



Advanced Simulation Technology inc.
500 A Huntmar Park Drive
Herndon, Virginia 20170 U.S.A.
Tel. (703)471-2104 • Fax. (703)471-2108
www.asti-usa.com

ASTi Telestra 4 ACE Studio Components Reference Guide

Document: DOC-01-TELAS-CRG-4

Product Name: Telestra 4 ACE Studio

ASTi ACE Studio Component Guide

© Copyright ASTi 2012-2014. ASTi documents are continuously updated at <http://support.asti-usa.com/>.

Restricted Rights: Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013.

This material may be reproduced by or for the U.S. Government pursuant to the copyright license under the clause at DFARS 252.227-7013 (1994).

ASTi

500 A Huntmar Park Drive

Herndon, VA 20170

Table of Contents

1.0. Introduction	1
1.1. Component Viewer	1
1.2. How to Use this Reference Manual	2
2.0. Audio Components	3
2.1. AmpMod	4
2.2. Audio Feed	6
2.3. AutoDred	7
2.4. Complex Playsound	12
2.5. Compressor	14
2.6. Delay	19
2.7. Demux	20
2.8. Envelope	21
2.9. Filter	23
2.10. Lockout	26
2.11. Level D Capture	28
2.12. MessageList	30
2.13. Mixer	32
2.14. Noise Source	34
2.15. PEnvelope	37
2.16. PFilter	38
2.17. Playsound	39
2.18. Pulse	41
2.19. Pulse Sequence	43
2.20. PulseStep	46

2.21. PulseStream	49
2.22. Record Replay	54
2.23. SimpleMixer	58
2.24. Sequencer	59
2.25. Volume Control	61
2.26. Vox	62
2.27. Wave	65
3.0. Audiolo	68
4.0. Comm Panel	69
4.1. Comm Panel 4, 8, 16, 32	69
4.2. CommPanel8Stereo	73
4.3. Stereo Comm Panel	76
5.0. Control Components	78
5.1. BitToByte	78
5.2. ByteToBit	79
5.3. ByteMerger	80
5.4. Byte Splitter	82
5.5. Counter	84
5.6. Delay	87
5.7. Ident	88
5.8. Incrementer	89
5.9. IntCompare	90
5.10. IntTable	92
5.11. Latch	93
5.12. LogicTable	94
5.13. MathFunction	96

5.14. NumToString	98
5.15. PassThrough	100
6.0. Dynamics	101
6.1. AGC	102
6.2. CompressorLimiter	104
6.3. Expander	106
6.4. Gate	108
7.0. Environmental Cue	110
7.1. 5BandFilter	110
7.2. Engine	113
7.3. Engine Level D	116
7.4. MultiFilter	118
7.5. Prop Rotor	121
7.6. Vibration Capture	123
8.0. Highway Service	124
8.1. AuralCue	125
8.2. AuralCuePosn	126
8.3. SpeakerOutput	127
9.0. Highway 3D Service	128
9.1. Audio > Audio Feed	131
9.2. Feeders > Aural Cue Posn	132
9.3. Feeders > Balancer1, 4,8,16	133
9.4. AudioIO > Headphone3DOut	134
9.5. AudioIO > HighwayOut	135
9.6. AudioIO > SpeakerOut	136

10.0. HRTF Service	137
10.1. HRTFOut4	138
10.2. CommPanel8HRTF4	140
11.0. IOInterfaces	142
11.1. ACE_RIU_channel	142
11.2. ACE_RIU_SerialByteOut	144
11.3. ACUchannel	145
11.4. ACU2channel	148
11.5. ACU2_SerialByteOut	151
11.6. AmpOut	152
11.7. SerialPort	153
11.8. VoisusChannel	154
12.0. Intercoms	157
12.1. IcomRx	157
12.2. IcomTx	159
12.3. Intercom Bus Power	160
12.4. Intercom Bus Service	161
13.0. Platform	162
13.1. Detonation	163
13.2. Fire	163
13.3. Geocentric Position	164
13.4. Geodetic Position	165
13.5. Relative Position	166
14.0. Host Control	171
14.1. HostIn	171

14.2. Host Out	175
14.3. Cell Service	176
CellIn	176
CellOut	178
15.0. Radio Components	179
15.1. ColocatedBeacon	180
15.2. Generic Control	183
15.3. HFServer	186
15.4. Intercom Transceiver	188
15.5. ICU	191
15.6. MarkerTone	192
15.7. MorseKeyer	196
15.8. RCUbasic	198
15.9. Receiver	200
15.10. Relay	202
15.11. Satellite	205
15.12. Transceiver	209
15.13. Transmitter	220
15.14. VORTAC_Controller	222
16.0. Speech	224
16.1. SpeechFeed	224
16.2. TextToSpeech	225
17.0. Remote Control	226
17.1. URC-200	226
Appendix A: Component Revision Notes	233

Appendix B: Component Credit Allocation 238

1.0. Introduction

ACE Studio is a powerful suite of software tools providing a software development toolkit for building sound and communications models. In ACE Studio, models are developed using an array of complex components.

By modifying the components, the user can construct anything from a small simulation element to a complete sound and communications audio modeling system for their application. In other words, the components are flexible and sufficiently configurable by the user to construct basic intercom systems to models that closely match the functionality of a commercial or military platform communications system.

1.1. Component Viewer

In ACE Studio, the user has the ability to double-click each component to open the component viewer. The component viewer provides specific component information and the component values. Each component has two views - the filter view, that only displays the most important parameters that must be set, and the unfiltered view, (full view) that displays every parameter available for that component.

Data – The data viewer tab lists the primitives in a tree view. The user can click on the arrows to expand the view to include the variables within a primitive. In some cases, a variable within a primitive contains its own set of variables and can also be expanded. The dotted lines in the 'From' and 'To' columns represent links into the component.

Links – The link inspector tab displays input and output links and details including Out Source Variable, Destination, Destination Variable, and In Source, Source Variable, and Destination Variable.

Schematic – The component schematic shows the processing logic within the component. In this view, the component building blocks known as primitives are shown in red or blue. Red indicates that the primitive handles audio while blue indicates that the primitive only handles control logic.

Info. – This may contain information on the component.

View/Edit Description – This allows the user to add a description about the component.

1.2. How to Use this Reference Manual

The purpose of this manual is to provide extensive information on the ACE Studio component structure and the operation of each component.

The components are organized in the following order.

- Audio Components
- AudioIO Components
- Comm Panel Components
- Control Components
- Dynamics Components
- Environmental Cue Components
- Highway 3D Service Components
- Intercom Components
- IOInterfaces Components
- Intercoms Components
- Platform Components
- Host Control Components
- Radio Components
- Speech Components

Note: Not all of the features and menu items that appear on your system will be described in this manual.

While the components allow the user to construct much of the audio simulation and infrastructure for a given application, it is still necessary for the user to develop a good portion of additional simulation code to drive the constructed model in sufficient fashion to fully realize the sound and communications operations for their application.

Each component is listed with a general summary and description and the remainder of the section is divided into table formats for the inputs, outputs, and internal parameters. For the remainder of this document each component section is organized into the following table sections:

- Inputs
 - Audio Inputs
 - Control Inputs
- Outputs
 - Audio Outputs
 - Control Outputs
- Internal parameters- These are any values that must be set as part of a component configuration that DO NOT have an external connection port.

Note: Not every component will have all the tables listed above, tables may vary depending on the complexity of the component.

Within each table the parameters are organized alphabetically for search ability.

2.0. Audio Components

These components can be mixed, filtered, or added into any combination of highway channels via a feeder connection.

The following section details the audio components and the parameters within them.

The audio components include:

- Ampmod
- Audio Feed
- Auto DRED
- Complex Playsound
- Compressor
- Delay
- Demux
- Envelope
- Filter
- Level D Capture
- Lockout
- MessageList
- Mixer
- Noise Source
- PEnvelope
- PFilter
- PlaySound
- Pulse
- PulseSequence
- PulseStep
- PulseStream
- Record Replay
- SimpleMixer
- Squencer
- Volume Control
- Vox
- Wave

2.1. AmpMod

Summary: The Amplitude Modulator component generates a carrier signal whose amplitude is controlled by a modulating signal. This is useful for general warning tones (e.g. Radar Warning Receivers) that require dynamic control. Complex warning tones can be generated when the amplitude modulator is used with a Pulse component.

Description: The amplitude modulator provides a signal multiplication capability between two signals, a carrier waveform and a modulating envelope.

The carrier waveform is the external signal source that is connected to the carrier input parameter. The CarrOffset input parameter is an offset value that is added to the amplitude of the carrier waveform prior to modulation. If no signal source is connected to Carrier and Carrier Offset is 0, the amplitude modulator does not generate an output signal. If no signal source is connected to Carrier and Carrier Offset is not 0, the amplitude modulator will use the offset value as a dc carrier (offset).

The modulating signal is the external signal source that is connected to the ModSignal input parameter. Typically, the modulating signal source is a square wave or pulse; however, any signal type may be used. The ModOffset input parameter is the offset value that is added to the amplitude of the modulating signal before the lag filter is applied. The modulation signal can thereby be offset from zero to allow for control of the modulation depth. The modulation offset should be 1.0 to provide a full depth of modulation from a square or sinusoidal source. This assumes the gain of the originating signal is set to 1.0, in which case it will swing between -1.0 and 1.0, hence the need for a 1.0 offset. The Modulator input flag controls whether the modulating signal is applied to the carrier. If not signal source is connected to ModSignal, the amplitude modulator does not modulate the carrier signal.

The FilterFreq internal parameter determines the filter constant for the lag filter, which filters modulating signal. This lag filter softens the edges which occur when square wave modulating a sine wave. The filter constant determines the effective slew rate of the modulating signal.

The Gain input parameter controls the amplitude of the playsound output. When the gain is less than or equal to 0, the amplitude modulator component does not output a signal.

Note: The Lag filter is an a-rate function, not a k-rate function.

Audio Inputs		
Carrier-Signal	Type	Default Value
	audio	n/a
	Description: The carrier signal whose amplitude is modulated.	
CarrOffset	Type	Default Value
	float32	1.0
	Description: Value added to the carrier signal prior to modulation. If no external variable is connected to ModOffset, the offset scale factor is used; otherwise, the offset is the scale factor times the output result of the external variable. Modifier: Multiply Modifier_default: 0.0 Range: 0.0 - 1.0	
Gain	Type	Default Value
	float32	1.0
	Description: Amplitude gain of the waveform. If no external variable is connected to Gain, the value of the scaler is used. Modifier: Multiply Modifier_default: 0.0 Range: 0.0-Inf	
ModulatingSignal	Type	Default Value
	audio	n/a
	Description: The modulating signal whose amplitude controls the amplitude of the output signal. It passes through a lag filter to round any sharp edges (as a square wave or pulse would produce) before it is applied to the carrier signal.	

Audio Inputs		
ModulationOffset	Type	Default Value
	float32	1.0
	Description: Value added to modulation signal prior to multiplication by carrier signal. If no external variable is connected to ModulationOffset, the offset scale factor is used; otherwise, the offset is the scale factor times the output result of the external variable. If a pulse stream is used then this offset should be set to 0.0 for an on/off modulation of the carrier. Modifier: multiply Modifier_default: 0.0 Range: 0.0-1.0	
Modulate	Type	Default Value
	boolean	false
	Description: Modulate control, when On carrier is modulated; otherwise, the carrier passes through with no modulation. If no external variable is connected to Modulate, the exclusive-or modifier is used as the local modulate flag; otherwise, the modulate value is the exclusive-or of the external variable and the modifier. Modifier: XOR Modifier_default: True	

Internal Parameters		
FilterFrequency	Type	Default Value
	float32	2400.0
	Description: Roll-off frequency for modulation signal lag filter (in Hertz). Modifier: Multiply Modifier_default: 2400.0	

Audio Outputs		
OutSignal	Type	Default Value
	audio	n/a
	Description: OutSignal is the output signal from the amplitude modulator component, which may be connected to another component or directed to an output highway.	

2.2. Audio Feed

This component is described in the Highways 3D Service section.

2.3. AutoDred

Summary: The AutoDred component conducts a series of five tests to test the system's speaker setup.

Description: The AutoDred component starts the first test by sending pink noise to all speakers in the system setup. Test 2 sends a sinewave at 100Hz out each individual speaker in the setup. Test 3 sends a sinewave at 1,000Hz out each individual speaker in the setup. Test 4 sends a sinewave at 10,000Hz out each individual speaker in the setup. Test 5 plays pink noise out each individual speaker.

If any of the tests fail, an Error Code will display the channel number that failed and the test number. If all tests pass, it confirms that all speakers are working properly in the system's setup.

Use the Manual Enable input to perform a specific test.

For the initial automated setup procedure, set the Test Enable and Setup Enable inputs to True. Set Manual Enable to False. This will run the setup tests for every channel. Setup will take approximately 2 minutes per channel, for 16 channels approximately 32 minutes.

Important: If possible, nobody should be in the room/simulator with the speakers during testing. If you must be in the same room during testing it is imperative that you remain still with no movement during testing.

Setup Procedure

This procedure assumes a prebuilt AutoDRED mode exists.

1. Install layout, open the AutoDRED model in load viewer.
2. Adjust AutoDRED gains (NoiseGain and FilteredNoiseGain) down to be careful at first. NoiseGain controls the level of the flat pink noise test. Filtered-NoiseGain controls the levels of the 3 other noise tests, which involve bandpass filtered pink noise.
3. Turn SetupEnable to TRUE, then TestEnable to TRUE to do a setup run. You should hear it cycle through four noises on each speaker. Wait for it to finish, then put SetupEnable and TestEnable back to FALSE.
4. SSH into the Target, view (using "less" or the editor of your choice) /tmp/autodred.dat, which has a nice summary of setup levels and distances above the noise floor for each test. The goal is for most tests to be >10 dB above the noise floor. This won't be possible on some tests, like high frequency noise through the subwoofers, and that is okay. Those cases will be considered "don't cares".
5. Adjust NoiseGain and FilteredNoiseGain, re-run the setup, and then reopen autodred.dat, until most tests reach the 10 dB mark. If one particular test (e.g. speaker 3, test 4) needs to be louder, the GainTable feature can be used to boost or cut that specific volume. Otherwise, GainTable does not need to be used.
6. When you are satisfied that most tests are 10 dB above the noise floor, create two TableXYs in the mathplan. Name one something like "DRED_Ref_Levels", and the other "DRED_Thresholds".
7. Enter the levels from /tmp/autodred.dat into the DRED_Ref_Levels table. See the example tables for how to structure the table. The test number goes along the top and the speaker number goes down the side.
8. Fill in the DRED_Thresholds table to have the value "3" (dB) in all entries that correspond to tests that were 10 dB or more above the noise floor. For test cases that were less than 10 dB above the noise floor, enter in a high threshold (for example 100) to make the case a "don't care".
9. Click "notify Target" in the Mathplan window to push the tables to the Target.
10. Select the two tables in the AutoDRED component - filling in "LevelTable" and "ThresholdTable".
11. Run the test a few times and observe the level differences on each test. Generally tests should be within 2 dB of the setup run.

Tips

- The 3 dB tolerances are adjustable to be tighter or looser depending on the requirements and what needs to be proved.
- The purpose of the /tmp/autodred.dat file is that it provides easier viewing of all the data. The same signal levels are shown in the component data viewer. Distance above noise floor is only shown in the file.
- The background noise test (test 5) can be useful, but it should only fail if there is a drastic change in background noise level. In the Mathplan tables for levels and thresholds, the background noise test is the first entry in the “5” column. To start, set a larger threshold, perhaps 6 dB.

TableXY Function									
Handle: H2_Ref_Levels					☐ Interpolate				
x vs y	1.0	2.0	3.0	4.0	5.0				
1.0	0.018577	0.02034	0.019079	0.018604	0.004785				
2.0	0.017907	0.020958	0.019826	0.01654	0.0				
3.0	0.021882	0.026971	0.026521	0.022479	0.0				
4.0	0.019454	0.022713	0.021153	0.018065	0.0				
5.0	0.022562	0.02875	0.028618	0.021359	0.0				
6.0	0.02385	0.034367	0.029084	0.01851	0.0				
7.0	0.019038	0.025189	0.018624	0.019144	0.0				
8.0	0.015582	0.019067	0.019626	0.015677	0.0				
9.0	0.014917	0.014911	0.027936	0.006797	0.0				
10.0	0.019935	0.019626	0.029523	0.007599	0.0				
11.0	0.022015	0.025077	0.024745	0.022565	0.0				
12.0	0.021411	0.041415	0.006172	0.004742	0.0				
13.0	0.014255	0.021929	0.005784	0.004764	0.0				

This TableXY should be populated with the measured RMS signal level for each speaker, for each test. These levels should be copied from the /tmp/autodred.dat file following a setup run.

TableXY Function									
Handle: H2_Thresholds					☐ Interpolate				
x vs y	1.0	2.0	3.0	4.0	5.0				
1.0	3	3	3	3	100.0				
2.0	3	3	3	3					
3.0	3	3	3	3					
4.0	3	3	3	3					
5.0	3	3	3	3					
6.0	3	3	3	3					
7.0	3	3	3	3					
8.0	3	3	3	3					
9.0	3	3	3	3					
10.0	3	3	3	3					
11.0	3	3	3	3					
12.0	3	3	3	3					
13.0	3	3	3	3					

Enter failure thresholds (in dB) into this TableXY for each speaker, for each test. The AutoDRED test will fail if one of the measured signal levels differs from the reference levels by more than the specified failure threshold.

Audio Inputs		
InSignal	Type	Default Value
	audio	n/a
	Description: Audio stream input links. All of the incoming audio is mixed in this component.	

Control Inputs		
Case Duration	Type	Default Value
	float32	0.0
	Description: The amount of time sounds are played for each test case.	
Num-Channels	Type	Default Value
	int32	0
	Description: Set this value to the number of speakers in the system setup. Limit is 16 channels.	
Setup Enable	Type	Default Value
	boolean	true
	Description: If set to true, the AutoDred component initiates the automated setup procedure. This may take a few minutes depending on the number of channels, approximately 2 minutes per channel.	
Test2Freq	Type	Default Value
	float32	100.00
	Description: Sets the frequency for test 2.	
Test3Freq	Type	Default Value
	float32	1,000.00
	Description: Sets the frequency for test 3.	
Test4Freq	Type	Default Value
	float32	10,000.00
	Description: Sets the frequency for test 4.	

Control Inputs		
TestCase Select	Type	Default Value
	uint8	0
	Description: Use this with manual enable to select the test. 1 - Sends pink noise to all speakers. 2 - Sends a sinewave at 100Hz to each individual speaker. 3 - Sends a sinewave at 1,000Hz to each individual speaker. 4 - Sends a sinewave at 10,000Hz to each individual speaker. 5 - Sends pink noise to each individual speaker.	
Test Enable	Type	Default Value
	boolean	false
	Description: Set to true to begin tests 1-5.	

Control Outputs		
Channel	Type	Default Value
	uint8	0
	Description: Assign each bus 1-16 to intercom buses in the model.	
Error-Code	Type	Default Value
	uint32	0
	Description: Displays the number of the speaker that fails. This number will appear until Test Enable is reset.	
LevelDifference	Type	Default Value
	Float32	0
	Description: Decibel difference between the last channel tested and the reference value that was created during the initial setup procedure.	
TestCase	Type	Default Value
	uint8	0
	Description: Use this with manual enable to specify the test. 1 - Sends pink noise to all speakers. 2 - Sends a sinewave at 100Hz to each individual speaker. 3 - Sends a sinewave at 1,000Hz to each individual speaker. 4 - Sends a sinewave at 10,000Hz to each individual speaker. 5 - Sends pink noise to each individual speaker.	

Control Outputs		
TestRunning	Type	Default Value
	boolean	False
	Description: The currently running test, if there is an error this field displays the test that failed. This test will appear until TestEnable is reset.	
Time	Type	Default Value
	float32	0.0
	Description: Amount of time the test has been running.	

Internal Parameters		
Background Level	Type	Default Value
	float32	0.0
	Description: The measured RMS level for the background noise test.	
Filter-NoiseGain	Type	Default Value
	float32	1.0
	Description: Filters the gain for pink noise tests.	
GainTable	Type	Default Value
	function	n/a
	Description: Select a TableXY from the Math Plan which is used to adjust the audio gain for each individual test. For example, this could be used to boost the low frequency noise volume for a sub-woofer test.	

Internal Parameters		
Level Table	Type	Default Value
	function	n/a
	Description: Select a TableXY from the Math Plan. This table should contain the measured signal levels for each test. These values can be copied from the /tmp/autodred.dat file following a setup run.	
Mode	Type	Default Value
	dred_mode	STOP
	Description: Reports the current state of the component. Options include: • Stop • Test • Manual Test • Setup	
NoiseGain	Type	Default Value
	int32	0
	Description: Controls the gain for pink noise tests.	
Result1	Type	Default Value
	float32	0
	Description: The output of Result1 is the audio level for all speakers at once.	

Internal Parameters		
Result2	Type	Default Value
	float32[16]	0
	Description: The output of Result2 is the audio level per speaker starting with 0.	
Result3	Type	Default Value
	float32[16]	0
	Description: The output of Result3 is the audio level per speaker starting with 0.	
Result4	Type	Default Value
	float32[16]	0
	Description: The output of Result4 is the audio level per speaker starting with 0.	
Result5	Type	Default Value
	float32[16]	0
	Description: The output of Result5 is the audio level per speaker starting with 0.	
Threshold Table	Type	Default Value
	function	n/a
	Description: Select a TableXY from the Math Plan. This table contains the failure thresholds for each individual test. Values are in dB.	

2.4. Complex Playsound

Summary: The Complex Playsound component plays digitally encoded soundfiles with dynamically varying elements. The elements occur in three sequences a ramp up sound, a loop file, and a run down sound.

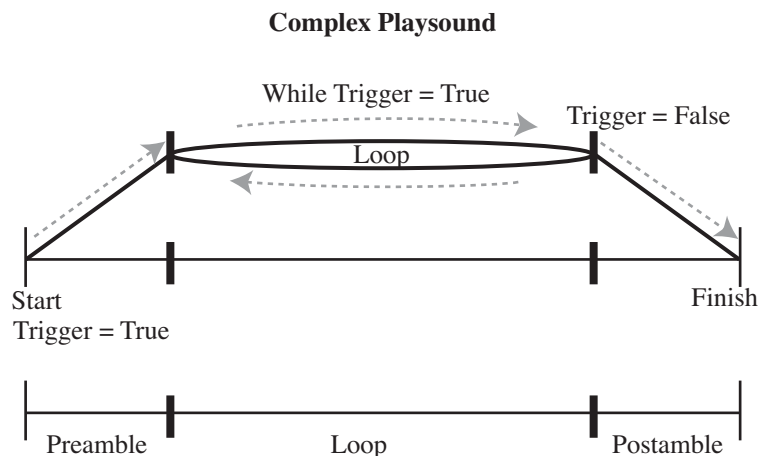
For those users familiar with the MBV sound library complex loop, this is essentially the same thing only now this function is provided by a separate component.

Note: Playsounds used in the Complex Playsound must be set to buffer for proper operation. In the Sound Library set the buffer to true for the sound.

Description: The Complex Playsound component starts playing the preamble soundfile when the trigger goes True. After playing the preamble in its entirety the loop soundfile is then played. The loop soundfile continues to play in a loop until the trigger goes False. At this point the postamble sound file is played.

The trigger and pause input parameters control playback. Within the complex playsound component, the library ID must be set. The group ID can be set locally within the component, and also modified or set by an external control. The indices (preamble, postamble, and loopsound) determine which soundfiles are used within the specified library and group. A group value of 0 indicates that the soundfile is not in a group, but directly under the library.

The output signal can be connected to any component that accepts audio as an input, such as a Balancer or a Mixer.



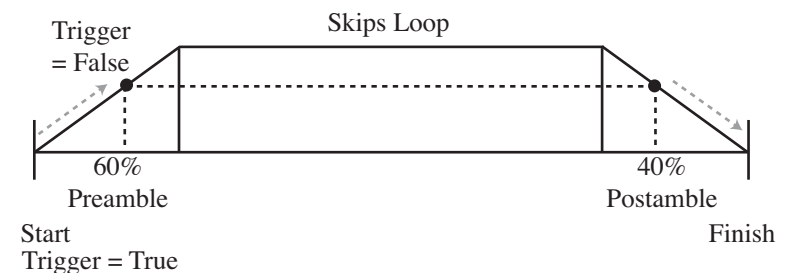
Audio Inputs		
LoopSoundIdx	Type	Default Value
	playsound_sound	0
	Description: The value of the loop file index to be played. This index is used to select a file from within a group. If no matches are found then no file is played. If no external variable is connected to Index, the offset is used.	
PostambleSoundIdx	Type	Default Value
	playsound_sound	0
	Description: The value of the postamble file index to be played. This index is used to select a file from within a group. If no matches are found then no file is played. If no external variable is connected to Index, the offset is used.	
PreambleSoundIdx	Type	Default Value
	playsound_sound	0
	Description: The value of the preamble file index to be played. This index is used to select a file from within a group. If no matches are found then no file is played. If no external variable is connected to Index, the offset is used.	
Trigger	Type	Default Value
	boolean	false
	Description: The trigger state, a value of TRUE starts playing the soundfile.	

Control Inputs		
GroupID	Type	Default Value
	playsound_group	0
	Description: The value of the GroupID. The GroupID is used to select a group from within a sound library. Modifier: Add Modifier_default: 0 Range: 0-255	
OutGain	Type	Default Value
	float32	1.0
	Description: OutGain applies amplitude gain control to the output signal. If no external control is connected to OutGain, the scale factor is used as the OutGain value.	
Pause	Type	Default Value
	boolean	false
	Description: The pause state, a value of TRUE freezes the soundfile playing, a value of FALSE allows the soundfile to continue from the current file position. If no external variable is connected to Pause, the modifier is used as the local value. Modifier: XOR Modifier_default: False	
Proportion Postamble Play	Type	Default Value
	boolean	false
	Description: The Proportional Postamble Play variable offsets the starting position of the postamble soundfile relative to the current position of the preamble soundfile. See diagram for an example.	

Audio Outputs		
Out	Type	Default Value
	boolean	False
	Description: Out is the output signal from the complex playsound component, which may be connected to another component.	

Internal Parameters		
LibraryID	Type	Default Value
	playsound_library	0
	Description: The Library ID selects a file from the sound library files.	

Proportional Postamble Play



Example: When the trigger equals True the preamble starts to play to the 60 percent position.

When the trigger equals false the postamble will then start playing at the 40 percent position.

2.5. Compressor

Summary: This component includes a gate, expander, compressor, and limiter module. The Compressor component modifies the dynamic range of an audio signal to provide a form of automatic volume control.

Description: The gate module acts to ensure that there is no audio signal below a certain threshold level. Once the signal level falls below the specified threshold, the gate will close after the hold time has passed and will output a signal with a gain of 0.

The expander module makes quiet sounds quieter. That is, if a signal is under the expander threshold, its gain is reduced by the specified ratio. Lower ratios produce more aggressive gain reductions.

The compressor module makes loud sounds quieter. If a signal rises above the compressor threshold the compressor will start to reduce the gain on that signal. The higher the ratio value is, the more aggressive this gain reduction will be.

The limiter module acts like a compressor with an infinite ratio and instant attack so that the output signal is a scaled version of the input signal with the highest peak level reaching the threshold.

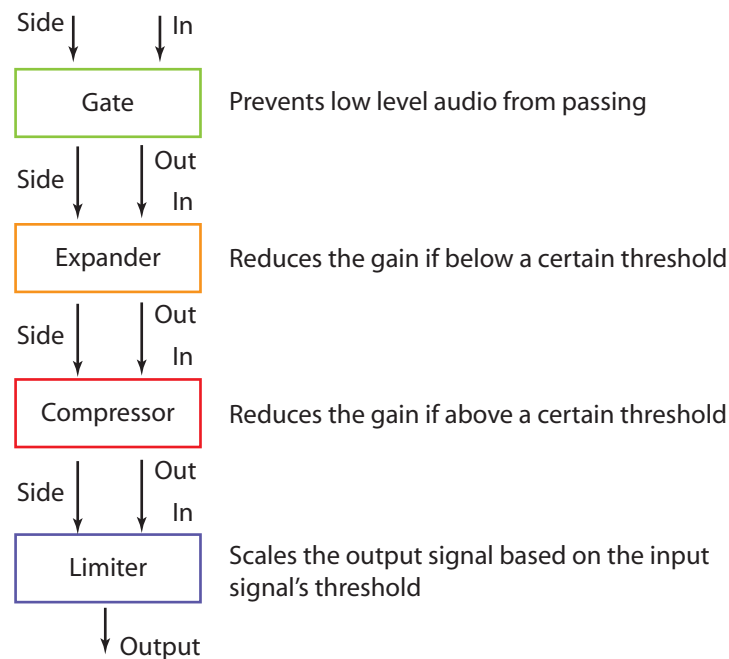


Figure 1: Compressor Process

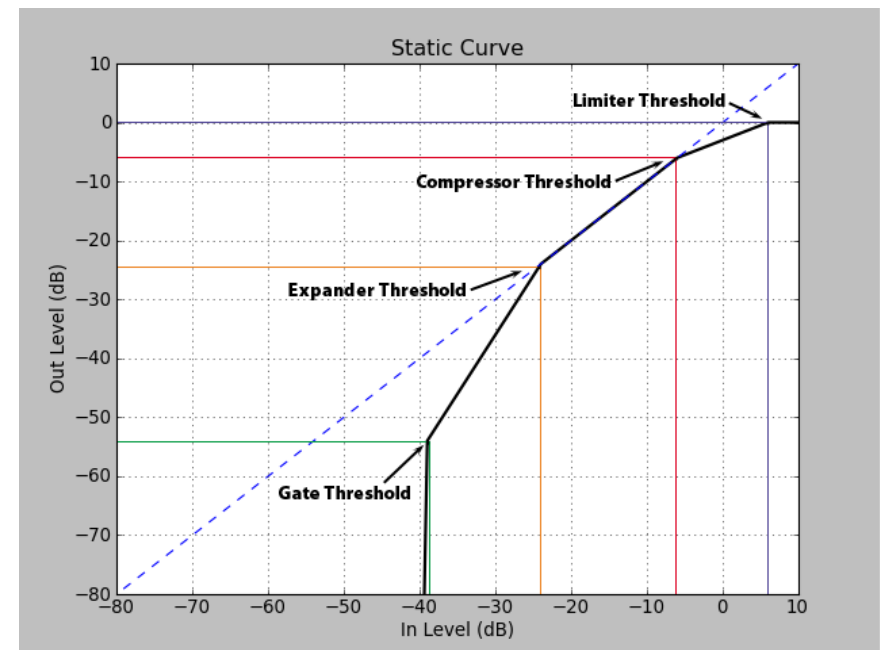


Figure 2: Compressor Static Curve

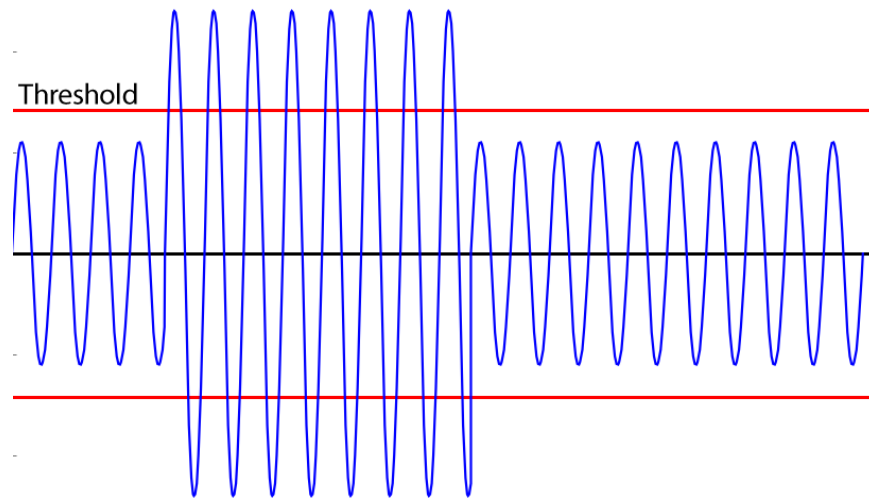


Figure 3: Compressor Input

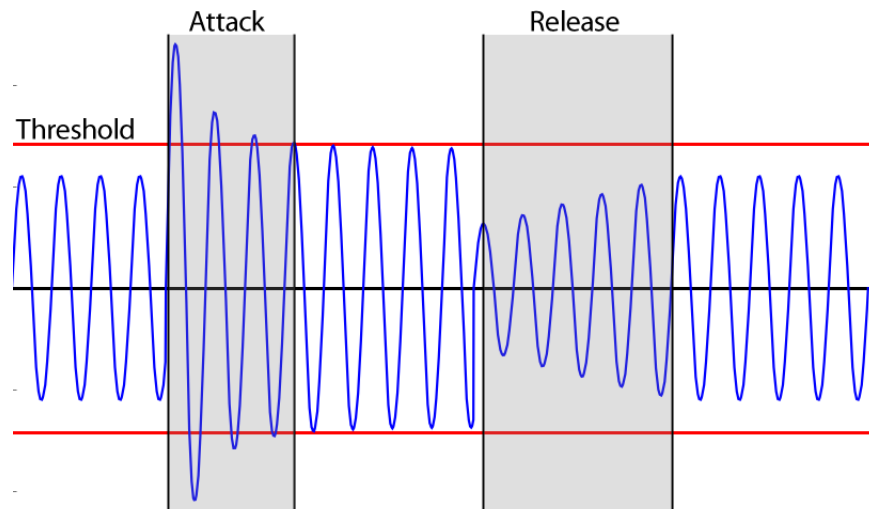


Figure 4: Compressor Output

Audio Input		
InSignal	Type	Default Value
	audio	n/a
	Description: Input signal to the Compressor component.	
SideIn	Type	Default Value
	audio	n/a
	Description: Side-chain input to the Compressor component.	

Control Input		
Gate	Type	Default Value
	Boolean	False
	Description: This control determines whether the signal will be gated (True) or passed through (False).	
Gate Side Enable	Type	Default Value
	Boolean	False
	Description: This control determines whether the envelope, with which the gate performs its gain control logic, will be based on the signal from SideIn (True) or InSignal (False).	
Gate-Threshold	Type	Default Value
	float32	-60.0
	Description: Any signal content below this level is muted. Values are in dB.	

Control Input		
GateRelease	Type	Default Value
	float32	100.00
	Description: This is the release time (in milliseconds) of the gate. The longer this time is, the longer the gate will remain open after the input signal has fallen below the threshold level.	
GateHold	Type	Default Value
	float32	200.00
	Description: This is the hold time (in milliseconds) of the gate. Once the envelope of the audio signal in question falls below the specified threshold, gate will activate after the hold time has elapsed.	
GateOutGain	Type	Default Value
	float32	1.0
	Description: This sets the linear gain factor for the output of the gate module. This gain is applied even if the gate is not enabled.	
ExEnable	Type	Default Value
	boolean	False
	Description: This control determines whether the signal will be expanded (True) or passed through (False).	
ExSideEnable	Type	Default Value
	Boolean	False
	Description: This control determines whether the envelope, with which the expander performs its gain control logic, will be based on the signal from SideIn (True) or InSignal (False).	

Control Input		
ExThreshold	Type	Default Value
	float32	-50.00
	Description: Any signal content below this level is expanded. Values are in dB.	
ExRatio	Type	Default Value
	float32	0.5
	Description: The Input to Output ratio applied once the input signal level falls below the expander threshold. For instance, with a ratio of 0.5 (1:2) if the input signal is below the threshold by 1 dB, the output will be below the threshold by 2 dB. Setting the ratio to a value greater than 1 will make the expander an 'upward expander'. Usable ranges for upward expanders are not much greater than 1 as they tend to boost signals dramatically.	
ExRelease	Type	Default Value
	float32	100.00
	Description: This is the release time (in milliseconds) of the expander. The longer this time is the longer it will take the expander to respond to falling signal levels.	
ExAttack	Type	Default Value
	float32	25.0
	Description: This is the attack time (in milliseconds) of the expander. The longer this time is the longer it will take the expander to respond to rising signal levels.	

Control Input		
ExOut-Gain	Type	Default Value
	float32	1.0
	Description: This sets the linear gain factor for the output of the expander module. This gain defaults to 1 if the expander module is not enabled.	
CompAttack	Type	Default Value
	float32	50.0
	Description: The amount of time it takes for gain reduction to take full effect when the input level exceeds the threshold. Values are in milliseconds.	
CompEnable	Type	Default Value
	boolean	True
	Description: This control determines whether the signal will be compressed (True) or passed through (False).	
CompSideEnable	Type	Default Value
	Boolean	False
	Description: This control determines whether the envelope, with which the compressor performs its gain control logic, will be based on the signal from SideIn (True) or InSignal (False).	
CompOutGain	Type	Default Value
	float32	1.0
	Description: This sets the linear gain factor for the output of the compressor module. This gain defaults to 1 if the compressor module is not enabled.	

Control Input		
CompRatio	Type	Default Value
	float32	4.0
	Description: The Input to Output ratio applied once the input signal level rises above the compressor threshold. For instance, with the default ratio of 4.0 (4:1) an input signal that is 4 dB above the threshold will yield an output signal that is only 1 dB above the threshold.	
CompRelease	Type	Default Value
	float32	500.0
	Description: The amount of time it takes for gain reduction to dissipate when the input level falls below the threshold. Values are in milliseconds.	
CompThreshold	Type	Default Value
	float32	-6.0
	Description: Any signal content above this level is compressed. Values are in dB.	
LimAllowClip	Type	Default Value
	Boolean	False
	Description: This control enables a naïve limiter algorithm that essentially acts to clip the signal if it breaks the specified threshold.	
LimEnable	Type	Default Value
	Boolean	False
	Description: This control determines whether the signal will be limited (True) or passed through (False).	

Control Input		
LimSi-deEnable	Type	Default Value
	Boolean	False
	Description: This control determines whether the envelope, with which the limiter performs its gain control logic, will be based on the signal from SideIn (True) or InSignal (False).	
LimOut-Gain	Type	Default Value
	float32	1.0
	Description: This sets the linear gain factor for the output of the limiter module. This gain is applied even if the limiter is not enabled.	
LimRe-lease	Type	Default Value
	float32	100.0
	Description: This is the release time (in milliseconds) of the limiter. The longer this time is the longer it will take for the effect of the limiter to wear off after the input signal level falls below the threshold.	
LimThres hold	Type	Default Value
	float32	0.0
	Description: This is the specified level that the signal will be limited to. Values are in dB.	

Audio Outputs		
OutSignal	Type	Default Value
	audio	n/a
	Description: The signal after being processed by the entire compressor component.	

Internal Parameters		
CompOut-Signal	Type	Default Value
	audio	n/a
	Description: The signal after being processed by the compressor module.	
GateOut-Signal	Type	Default Value
	audio	n/a
	Description: The signal after being processed by the gate module.	
ExOutSig-nal	Type	Default Value
	audio	n/a
	Description: The signal after being processed by the expander module.	

2.6. Delay

Summary: Delays the input audio.

Description: The Delay component delays audio by the specified number of frames (1.33 ms) up to one second.

Audio Inputs		
AudioIn	Type	Default Value
	audio	n/a
	Description: The audio signal that is to be delayed.	

Control Inputs		
Delay-Frames	Type	Default Value
	int32	0
	Description: The number of frames to delay the input audio. Each fame equals 1.33 ms.	
Enable	Type	Default Value
	Boolean	False
	Description: When True the audio signal is delayed.	
Gain	Type	Default Value
	Float32	1.0
	Description: The audio signal's strength in volume.	

Internal Parameters		
DelayMS	Type	Default Value
	Float32	0
	Description: The frame delay expressed in milliseconds. DelayFrame x 1.33 ms	

Audio Inputs		
AudioOut	Type	Default Value
	audio	n/a
	Description: The delayed audio signal.	

2.7. Demux

Summary: The Demultiplexer (Demux) component takes an input signal and divides it among 15 outputs as specified by the control.

Description: The Demux is made up of one signal input, one control input and 15 signal outputs.

The Demux is commonly used for environmental cue applications where it is necessary to test different speakers in the simulator.

Audio Inputs		
InSignal	Type	Default Value
	audio	n/a
	Description: Provides a connection to an audio signal that is to be distributed to the signal outputs.	

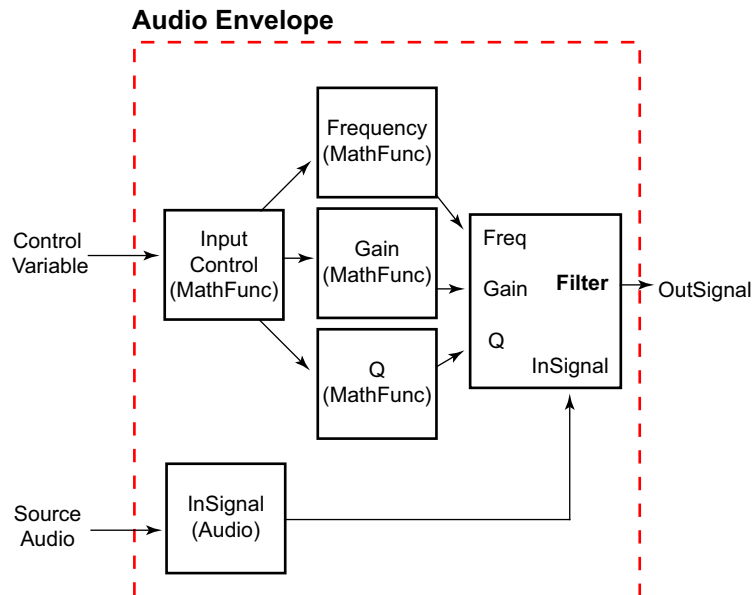
Control Inputs		
Control	Type	Default Value
	int32	0
	Description: Specifies which of the 15 OutSignals receive the audio signal. The default of 65535 sends the audio signal to all 15 outputs.	

Audio Outputs		
OutSignal 0 - OutSignal 15	Type	Default Value
	audio	n/a
	Description: OutSignal0 - OutSignal15 are the output signals from the Demux component. The audio signal can be routed out to 1 out-signal or up to 15 outsignals as designated by the control input.	

2.8. Envelope

Summary: The Envelope component applies a filter to an input signal based on three math functions.

Description: The Envelope component applies a type of filtering to the input signal. The type of filtering may be lowpass, bandpass, or highpass. The input signal's frequency, gain, and Q factor are calculated in the user-defined Control/ Math Function component.



Audio Inputs		
InSignal	Type	Default Value
	audio	n/a
Description: Provides a connection to an audio signal for filtering.		

Control Inputs		
Control	Type	Default Value
	float32	1.0
Description: Determines what the input is going to be based on the math functions.		
Enable	Type	Default Value
	boolean	False
Description: When True the filter is turned on.		

Audio Outputs		
OutSignal	Type	Default Value
	audio	n/a
Description: The output audio signal after it is filtered.		

Internal Parameters		
FilterType	Type	Default Value
	FilterType	n/a
Description: Defines the filter type including, lowpass, bandpass, and highpass.		
Frequency	Type	Default Value
	n/a	n/a
Description: Defined in Control/ Math Function to set the audio signal's frequency.		

Internal Parameters		
Gain	Type	Default Value
	n/a	n/a
	Description: Defined in Control/ Math Function to set the audio signal's gain.	
Q	Type	Default Value
	n/a	n/a
	Description: Defined in Control/ Math Function to set the audio signal's Q.	

2.9. Filter

Summary: The Filter component applies a filter to an input signal. The type of filtering applied may be lowpass, bandpass, or highpass. The filter quality factor, rolloff frequency, and gain can be controlled by input variable from elsewhere in the model or from the host interface.

Description: The Filter component applies to an input signal. The type of filtering applied may be lowpass, bandpass, or highpass. The filter quality factor, rolloff frequency, and gain can be controlled by input variables from elsewhere in the model or from the host interface.

The input signal audio should be passed to the filter only if the derived output gain is greater than 0.0.

The filter is controlled by the parameters: Enable, Type, Filter Frequency, and Q Factor. The Filter Enable control is specified by an externally controlled variable and an XOR logic gate control. XOR logic is applied to the external variable and the gate control to derive a final value for the Filter Enable flag. If no external variable is connected, the value of the logic gate control becomes the Filter Enable flag. If Filter Enable is set to False, no filtering is applied to the white noise.

Type determines the type of filter applied to the input signal (Off, Lowpass, Highpass, or Bandpass). If Type is set to Off, no filtering should be applied to the white noise. Filter Frequency (in Hertz) provides the rolloff frequency of the filter. The meaning of the frequency depends on the filter type as follows:

Filter TypeFrequency Meaning

OffIgnored

LowpassUpper Bound Frequency

HighpassLower Bound Frequency

BandpassCenter frequency with bandwidth controlled by Q factor (see below)

Q Factor is the filter quality factor, which effectively determines the filter rolloff for lowpass and high pass filters and the filter bandwidth for bandpass filters. Q factor is inversely proportional to the filter rolloff or bandwidth.

The filter parameters are used to derive the filter coefficients for a 2-pole IIR filter. If Q Factor is 0 or less, a very low non-zero value should be used for the k calculation. The output from the filter is active only if the white noise source is active. If the Filter Type is Off, the filter output signal is the input signal.

An amplitude gain control is applied to the filter output. This output gain control consists of a connection to an external control variable and a scale factor. The value of the external variable is multiplied by the scale factor to drive the output gain. If

no external variable is connected, the value for the scale factor becomes the output gain.

The Filter component output audio should only be active if the following conditions are true:

- I. The output from the filter is active.
- II. The derived output gain (external variable * scale factor) is greater than 0.0.

Audio Inputs		
InSignal	Type	Default Value
	audio	n/a
	Description: InSignal is the connection to the signal used as an input to the filter.	

Control Inputs		
Enable	Type	Default Value
	boolean	False
	Description: FilterEnable is a flag that determines if the white noise signal is filtered. If no external control is connected to FilterEnable, the xor modifier is used as the FilterEnable value. Modifier: XOR Modifier_default: True	
Frequency	Type	Default Value
	float32	1.0
	Description: Frequency (in Hertz) provides the rolloff frequency of the filter. For lowpass filters, Frequency acts as an upper bound frequency. For highpass filters, Frequency acts as a lower bound frequency. For bandpass, Frequency acts as a center frequency and the filter bandwidth is determined by Qfactor. If no external control is connected to Frequency, the scale factor is used as the Frequency value. The acceptable range is 0 to half of the model sample rate. Modifier: Multiply Modifier_default: 2000.0	

Control Inputs		
OutGain	Type	Default Value
	float32	1.0
	Description: OutGain applies amplitude gain control to the output signal. If no external control is connected to OutGain, the scale factor is used as the OutGain value. Modifier: Multiply Modifier_default: 1.0	
Qfactor	Type	Default Value
	float32	1.0
	Description: Qfactor is the filter quality factor, which effectively determines the filter rolloff for lowpass and highpass filters and the filter bandwidth for bandpass filters. Qfactor is inversely proportional to the filter rolloff or bandwidth. The scale factor is used as the Qfactor value. Modifier: Multiply Modifier_default: 0.7071	

Audio Outputs		
OutSignal	Type	Default Value
	audio	n/a
	Description: OutSignal is the output signal from the Filter component.	

Internal Parameters		
FilterType	Type	Default Value
	filter_type2	Lowpass
	Description: FilterType determines the type of filter applied to the input signal (Lowpass, Highpass, Bandpass, LowPassQ, BandpassQ, HighPassQ, and Notch). If no filtering is desired, FilterType should be set to Off.	

2.10. Lockout

Summary: The Lockout Component allows sharing of a limited number of assets on a "first-come, first-served" basis.

Description: This component routes input audio streams (referred to as Operators) to output audio streams (referred to as Assets) based on when each input triggers their PTT. If the number of Input streams is greater than the available outputs, those who PTT last are unable to get access to an output and are "locked out".

The Lockout Component accepts up to 8 separate Operator Inputs. Operators are then assigned to the Output Assets based on when they PTT. The first operator to PTT will gain access to Asset 1, the second Asset 2, etc. If the number of available Assets is exceeded, the next Operator in will be locked out and unable to gain access to any of the Assets.

Once an operator releases their PTT, the Asset which was allocated to them will become available to all operators once the Release Time elapses. The component is similar to an office phone system, whereby multiple outside lines are accessible by many people, but the number of simultaneous calls is limited.

Audio Inputs		
Operator1 _Audio thru Operator8 _Audio	Type	Default Value
	audio	n/a
	Description: Input audio stream that corresponds to the PTT of the same name. Operator audio is routed to Asset Audio based on when the PTT is pressed. The audio stream is always routed to the lowest numbered Asset that is available.	

Control Inputs		
Asset Mask	Type	Default Value
	byte	255
	Description: The Asset Mask controls what Assets can be used by the Operator streams with the Least Significant Bit corresponding to Asset1, and the Most Significant corresponding to Asset8. For example, an Asset Mask of 15 would make Asset1, Asset2, Asset3, and Asset4 available for the operators to use.	
Operator1 _PTT thru Operator8 _PTT	Type	Default Value
	boolean	False
	Description: The Operator PTT acts as a trigger for each of the 8 input streams. A value of TRUE should be used when an Operator is trying to gain access to an Asset. The Operator PTT can also be used without the Operator Audio.	
Operator Mask	Type	Default Value
	byte	255
	Description: The Operator Mask controls what Operators can be routed to the Asset streams with the Least Significant Bit corresponding to Operator1, and the Most Significant corresponding to Operator8. For example, an Operator Mask of 31 would make Operators 1, 2, 3, 4, and 5 available for use within the component.	
Release Delay	Type	Default Value
	float32	1.0
	Description: The time, in seconds, that the component waits after a PTT goes false before allowing an Asset to be used by another Operator.	

Audio Outputs		
Asset1_Audio thru Asset8_Audio	Type	Default Value
	audio	n/a
	Description: Operator audio that has to be routed out of the component once a PTT has been pressed. When a PTT goes true, the audio is always routed to the lowest available Asset.	

Control Outputs		
Asset1_Operator thru Asset8_Operator	Type	Default Value
	uint8	0
	Description: Reports what input audio stream (operator) is assigned to the corresponding output (Asset). For example, if Asset1_Operator is equal to 4, that indicates that the audio from Operator4_Audio is being routed to Asset1.	
OpAssigned-Mask	Type	Default Value
	byte	0
	Description: This mask reports if an operator is currently assigned to an asset. For example, a value of 3 indicates that Operator1 and Operator2 are both being routed to one of the Asset outputs.	

2.11. Level D Capture

Summary: The Level D Capture component is used to record audio for Level D compliance testing. The audio recorded by this component can be compared spectrally to the reference audio, in real time with the scope or offline with the spectral analysis capability accessible through RMS.

Description: To record audio using the Level D Capture component, link the record audio coming in through the ACU to the Level D Capture component. In the sound library add the aircraft reference recordings, set the path value and index number. In the Level D Capture component set the test number to match the file's index number, this selects the reference file to run with the test recordings. Record the audio. The recording then goes to a file on the Target. The file number will look something like this:

```
library000_group65535_index00 "x"
```

The '65535' signifies the Level D recording file. The 'X' is the test number.

Host control can direct the test number and the start variable.

In the Level D Capture component:

1. Select the library and group object (these are set in the Sound Library)
2. Set the ACU gain level to match the level of the reference recording.
3. Add the table function to calculate the gain level with the calibration number. (This is the number calculated during calibration of the ACU with the microphone.)
4. Set the 'x' to the test case number and the 'f(x)' to the calibration number. This value is sent to the ACU.
5. Set the 'Record Length' function for each test case to match the amount of time the file was recorded.
6. Set the maximum record time.

After recording the audio, open RMS and navigate to the Audio > Spectral Analysis page. Select the wavset that contains the reference files and RMS will automatically link up the reference file to the recorded file. Check the box to compare the files and/or create a difference plot to show the margin of the difference. Click the 'Start Spectral Analysis' button and RMS generates a .pdf file of the results.

Audio Inputs		
RecAudio	Type	Default Value
	boolean	False
	audio	n/a
	Description: The input is the audio typically coming in from a microphone linked to an ACU. The audio is recorded. Note: Similar to the Record Replay component.	

Control Inputs		
ACUGain	Type	Default Value
	function	n/a
	float32	1.0
	Description: Insert a table function containing the ACU gain corresponding to the current test number.	
RefAudio	Type	Default Value
	playsound_library	0
	playsound_group	0
	playsound_sound	1
	Description: The reference audio for comparison. Playsound_LibraryID: Selects the library from the soundlibrary containing the reference audio file to be played. Playsound_GroupID: Selects a group from the soundlibrary containing the reference audio file to be played. Playsound_SoundID: Selects a sound file from the soundlibrary. Note: Similar to the Playsound component.	

Control Inputs		
Start	Type	Default Value
	boolean	false
	Description: Triggers the playback of the reference file and the recording of RecAudio.	
TestNum	Type	Default Value
	uint16	1
	Description: Drives the index number of the reference file and the record file. Use one test number per test case.	

Audio Outputs		
PSAudio Out	Type	Default Value
	boolean	False
	audio	n/a
Description: The reference file audio for the specified test number.		

Control Outputs		
ACUGain Result	Type	Default Value
	function	n/a
	float32	1.0
	Description: The gain that is applied to the ACU input audio. Link out to the ACU channel component.	
RecState	Type	Default Value
	player_state	stopped
	Description: Displays the record state of the component when start goes to true, this value will change from “stopped” to “recording.”.	

Internal Parameters		
Record Length	Type	Default Value
	function	n/a
	float32	1.0
	uint32	120
	Description: Function: Insert a table function that contains the record time corresponding to the current test number. Result: The length of the recording. Max.: Use to set the maximum record length in seconds.	

2.12. MessageList

Summary: Plays through a list of sound files and is commonly used for sequencing ATIS or Warning messages.

Description: MessageList is similar to Playsound but accepts an array of sound indices to play. The sound indices are optionally set locally or driven by a host computer. Sounds are played sequentially. The next sound starts immediately after the current sound stops.

Control Inputs		
LibraryID	Type	Default Value
	playsound_sound	0
	Description: Sound library selection from the layout's sound repository.	
Group Index	Type	Default Value
	playsound_group	0
	Description: Selects a group from within the sound library.	
Sound Indices	Type	Default Value
	message	[0,0,0....0]
	Description: The array of sound indices to be played (up to 256).	
Trigger	Type	Default Value
	boolean	False
	Description: The current trigger state. A value of true starts the MessageList playing	

Control Inputs		
Reset	Type	Default Value
	boolean	False
	Description: When true, the MessageList stops playing. When set back to false, the MessageList resumes playing at the first sound index.	
Pause	Type	Default Value
	boolean	False
	Description: A value of true pauses playback of the current sound. When set back to false, play continues from the current file location.	
PlayAll	Type	Default Value
	boolean	False
	Description: Determines MessageList behavior when dettriggered. A value of true forces the entire list to play before stopping, a value of false allows the MessageList to stop immediately.	
Continuous	Type	Default Value
	boolean	False
	Description: Determines the MessageList behavior when the Trigger is held TRUE and the end of the list is reached. If the value is TRUE, MessageList loops back to the beginning of the list and continues playing. If the value is FALSE, MessageList will stop.	
OutGain	Type	Default Value
	float32	1.0
	Description: Applies amplitude gain control to the output signal. If no external control is connected, the scale factor is used as the gain value.	

Control Outputs		
Current-Sound	Type	Default Value
	playsound_sound	0
	Description: Outputs the currently playing sound index.	

Audio Outputs		
OutSignal	Type	Default Value
	audio	n/a
	Description: The output signal from the component.	
Repeat-Count	Type	Default Value
	uint16	n/a
	Description: This number increments each time the same index plays in a row.	

Internal Parameters		
Delay	Type	Default Value
	float32	0
	Description: The length of time (in seconds) to wait before looping back to the beginning of the list when in continuous mode.	

2.13. Mixer

Summary: The mixer component provides controlled mixing of up to eight signals into a single, composite signal. The mixer controls determine which of the eight signals should be mixed with both individual and overall gain control. There is also a ninth signal that is always mixed into the output signal and allows multiple mixer components to be cascaded together.

Description: The mixer component mixes up to nine signal sources into a single, composite output stream. The mixing of the first eight signals is controlled by the Control variable, which determines which of the eight signals is included in the output. Control is a single byte in which each bit controls the switch state of a signal. If a given bit is set, the signal corresponding to that bit will be added to the mixer output. The ninth signal, if present, is always added to the mixer output regardless of the value of Control.

Each of the nine signal sources has its own amplitude gain control. Each signal gain control consist of a connection to an external controlled variable and a scale factor. The value of the external variable is multiplied by the scale factor to derive the signal gain. If no external variable is connected, the value of the scale factor becomes the signal gain.

The amplitude of the final mixer output is controlled by the output gain control. Like the signal gains, the output gain is an external variable multiplied by a scale factor.

The mixer output audio should be active only if all of the following conditions are true:

- I. At least one if the signal sources (Signal 1-8 or InSignal) is active.
- II. The derived amplitude gain(s) (external variable * scale factor) for the active signal source(s) is greater than 0.0.
- III. The bit in the Control variable that corresponds to the active signal source is set. (Note: This condition does not apply to InSignal because it is not affected by the control variable).
- IV. The final mixer output gain control is greater than 0.0.

Audio Inputs		
InSignal	Type	Default Value
	boolean	False
	audio	n/a
	Description: Additional signal that is added to output of mixer and is independent of the control variable. This field allows multiple mixers to be cascaded together and thereby provides audio mixing of more than eight signals.	
InSig_Gain	Type	Default Value
	float32	1.0
	Description: Amplitude gain control for InSignal. If no external variable is connected to InSig_Gain, the value of the scaler is used. Modifier: Multiply Modifier_default: 1.0	
Out_Gain	Type	Default Value
	float32	1.0
	Description: Amplitude gain control for final mixer output. If no external variable is connected to Out_Gain, the value of the scaler is used. Modifier: Multiply Modifier_Default: 1.0	
Signal 1-8	Type	Default Value
	audio	n/a
	Description: The eight signals which are mixed into a composite output according to the control selectors and gain values.	

Audio Inputs		
Sig1_Gain- Sig8_Gain	Type	Default Value
	float32	1.0
	Description: The amplitude gain control for signals 1 through 8. If no external variable is connected to the signal gain, the value of the scaler is used. Modifier: Multiply Modifier_default: 1.0	

Audio Outputs		
OutSignal	Type	Default Value
	boolean	False
	audio	n/a
Description: OutSignal is the composite output from the mixer component, which may be connected to another component or directed to an output highway.		

Control Inputs		
Control	Type	Default Value
	uint8	255
	Description: Control switches the eight signals on or off. Each bit in the control byte enables one of the eight signals. When the least significant bit (i.e. Bit 0) is 1, signal 1 is added to the mixer output. The remaining signals are controlled by each bit in sequence, with the most significant bit (i.e. Bit7) being the control for signal 8. If no external variable is connected to Control, the value of the modifier is used. Modifier: AND Modifier_default: 255 Range: 0-255	

2.14. Noise Source

Summary: The Noise Source component generates a filtered white, pink, or brown noise signal. The type of filtering applied may be lowpass, bandpass, or highpass. The filter quality factor, rolloff frequency, and gain can be controlled by input variables from elsewhere in the model or from the host interface. The noise signal is an internal pseudo random noise source, providing an improved noise source with better tunability.

Description: This component consists of a signal source and a filter. The source signal on which the filter acts is white gaussian, pink, or brown noise. The noise signal should be active only if the derived OutGain is greater than 0.0.

The filter is controlled by the parameters: Enable, Type Filter Frequency, and Q factor. The Filter Enable control is specified by an externally controlled variable and an XOR logic gate control. XOR logic is applied to the external variable value and the gate control to derive a final value for the Filter Enable flag. If no external variable is connected, the value of the logic gate control becomes the Filter Enable flag. If Filter Enable is set to False, no filtering is applied to the noise signal.

Type determines the type of filter applied to the input signal (Off, Lowpass, Highpass, or Bandpass). If Type is set to Off, no filtering should be applied to the noise. Filter Frequency (in Hertz) provides the rolloff frequency of the filter. The meaning of the frequency depends on the filter type as follows:

Filter TypeFrequency Meaning

OffIgnored

LowpassUpper bound frequency

HighpassLower bound frequency

Bandpass Center frequency with bandwidth controlled by Q Factor (See below)

Q Factor is the filter quality factor, which effectively determines the filter rolloff for lowpass and highpass filters and the filter bandwidth for bandpass filters. Q Factor is inversely proportional to the filter rolloff or bandwidth.

The filter parameters are used to drive the filter coefficients for a 2-pole IIR filter.

If Q Factor is 0 or less, a very low non-zero value should be used for the k calculation. The output from the filter is active only if the noise source is active. If the Filter Type is Off, the filter output signal is the input signal.

An amplitude gain control is applied to the filter output. This output gain control consists of a connection to an external controlled variable and a scale factor. The value of the external variable is multiplied by the scale factor to derive the output gain. If no external variable is connected, the value for the scale factor becomes output gain.

The Noise Source component output audio should only be active if the following conditions are true:

- I. The output from the filter is active.
- II. The derived output gain (external variable * scale factor) is greater than 0.0.

Control Inputs		
FilterEnable	Type	Default Value
	boolean	FALSE
	Description: FilterEnable is a flag that determines if the white noise signal is filtered. If no external control is connected to FilterEnable, the XOR modifier is used as the FilterEnable value. Modifier: XOR Modifier_default: True	

Control Inputs		
FilterFrequency	Type	Default Value
	float32	1.0
	Description: FilterFrequency (in Hertz) provides the rolloff frequency of the filter. For lowpass filters, FilterFreq acts as an upper bound frequency. For highpass filter, FilterFreq acts as a lower bound frequency. For bandpass, FilterFreq acts as a center frequency and the filter bandwidth is determined by FilterQFactor. If no external control is connected to FilterFrequency, the scale factor is used as the FilterFrequency value. The acceptable range is 0 to half of the model sample rate. Modifier: Multiply Modifier_default: 2000.0	
FilterQ-Factor	Type	Default Value
	float32	1.0
	Description: FilterQFactor is the filter quality factor, which effectively determines the filter rolloff for lowpass and highpass filters and the filter bandwidth for bandpass filters. FilterQFactor is inversely proportional to the filter rolloff or bandwidth. This field cannot be less than or equal to 0. If no external control is connected to FilterQFactor, the scale factor is used as the FilterQFactor value. Modifier: Multiply Modifier_default: 0.7071	

Control Inputs		
Outgain	Type	Default Value
	float32	1.0
	Description: OutGain applies amplitude gain control to the output signal. If no external control is connected to OutGain, the scale factor is used as the OutGain value. Modifier: Multiply Modifier_default: 1.0	

Audio Outputs		
OutSignal	Type	Default Value
	boolean	true
	audio	n/a
Description: The filtered audio signal. If the filter type is set to off, it will not produce audio.		

Internal Parameters		
FilterType	Type	Default Value
	filter_type2	LowPass
	Description: FilterType determines the two pole filter type applied to the input signal (Lowpass, Highpass, Bandpass, LowPassQ, BandPassQ, HighPassQ, and Notch). If no filtering is desired, FilterType should be set to Off. The three Q filters are amplitude adjusted such that the filter has unity gain at the roll-off frequency, and maintains this gain as the quality factor is increased. The bandpass filters have the lowpass and highpass poles at the same roll-off frequency.	
NoiseColor	Type	Default Value
	noise_color	White
	Description: NoiseColor determines the type of noise applied to the input signal. White Noise: Noise with a flat power spectral density, the intensity is the same at all frequencies within a given band. Pink Noise: A random signal whose amplitude decreases as frequency increases, preserving constant audio strength per frequency increment. Brown Noise: Noise where the spectral density is proportional to $1/f^2$, it has more energy at lower frequencies, even more than pink noise.	

2.15. PEnvelope

Summary: The Parametric Envelope (PEnvelope) component applies a parametric filter (Pfilter) to an input signal based on three math functions.

Description: The PEnvelope component applies the internal band-pass filter to a given band of the input signal which is determined by three parameters: the center frequency, the bandwidth and the desired gain.

Audio Inputs		
InSignal	Type	Default Value
	audio	n/a
	Description: Provides a connection to an audio signal for filtering.	

Control Inputs		
Control	Type	Default Value
	float32	1.0
	Description: Determines what the input is going to be based on the math functions.	
Enable	Type	Default Value
	boolean	False
	Description: When True the filter is turned on.	
InputControlFunction	Type	Default Value
	boolean	False
	Description: Defined in Control/ Math Function to define the Input-ControlFunction.	

Audio Outputs		
OutSignal	Type	Default Value
	audio	n/a
	Description: The output audio signal after it is filtered.	

Internal Parameters		
Bandwidth	Type	Default Value
	n/a	n/a
	Description: Defined in Control/ Math Function to set the audio signal's bandwidth in hertz.	
FilterType	Type	Default Value
	FilterType	n/a
	Description: Defines the filter type including, lowpass, highpass, and bandpass.	
Frequency	Type	Default Value
	n/a	n/a
	Description: Defined in Control/ Math Function to set the audio signal's frequency.	
Gain	Type	Default Value
	n/a	n/a
	Description: Defined in Control/ Math Function to set the audio signal's gain.	

2.16. PFilter

Summary: The Parametric Filter (PFilter) component is a parametric equalizer that allows the user to control the amplitude of a given band of the input signal.

Description: The Pfilter component applies the internal band-pass filter to a given band of the input signal which is determined by three parameters: the center frequency, the bandwidth and the desired gain.

Audio Inputs		
InSignal	Type	Default Value
	audio	n/a
	Description: InSignal is the connection to the signal to be used as an input to the filter. The input signal comes from another component in the model.	

Control Inputs		
Bandwidth	Type	Default Value
	float32	1.0
	Description: Sets the bandwidth around its center frequency. Units are in Hertz.	
Enable	Type	Default Value
	boolean	true
	Description: When True it triggers the filter on.	
Frequency	Type	Default Value
	float32	1.0
	Description: The center frequency. Units are in Hertz.	

Control Inputs		
Gain_dB	Type	Default Value
	float32	1.0
	Description: Sets the gain value for the filter. Units are in decibels (dB).	

Audio Outputs		
OutSignal	Type	Default Value
	audio	n/a
	Description: The filtered audio signal. If Enable is set to false, it will pass original audio without filtering.	

2.17. Playsound

Summary: The playsound component provides the ability to play digitally encoded soundfiles. Sounds that have no dynamically varying elements (except for overall volume level) are best handled as fixed off-line recorded sound files (e.g. Missile launch).

Host inputs are provided to trigger playback, pause playback, set the start of file offset, set the end of file offset, adjust the overall output gain, and set the file index number for grouped sound files.

Description: The Trigger and Pause input parameters control playback. The playback behavior is determined by the Simple Loop and PlayAll flags that are configured as part of the sound file definition outside of the playsound component. If the sound file is configured for Simple Loop, the playsound plays the file to its end, then starts at the beginning again until the trigger is removed. If the sound file is configured for Play All, the playsound plays the file to its end when the trigger is removed.

If the playsound component points to a sound group or library, the Index input parameter determines which sound file within the group or library is played. The Index value must be greater than 0 for a sound file within a sound group to be played. The playsound does not play if Index is 0. If the playsound component points to a sound file, Index is ignored. If the trigger value is held true, and the index is incremented, the sound file will be played as if the trigger were set true with the new Index value. If the trigger is held true and the Index is incremented before the current file is done playing, it's behavior will be determined by the Playall and the Loop flags, as explained above.

There are several levels of organization for sound files. A Library is a collection of Groups. A group is a collection of sounds or playfiles. Within the playsound component, the Library ID must be set, and is not modified by an external control (for example a host control). The group ID can be set locally within the component, and also modified or set by and external control. The Index determines which soundfile within the specified Library and Group is selected. A Group value of 0 indicates that the soundfile is not in a Group, but directly under the Library (equivalent to Model Builder organization).

The Gain input parameter controls the amplitude of the playsound output.

The output signal from a pre-recorded sound file can be connected to any component that accepts audio as an input, such as a Mixer.

Control Inputs		
Begin Offset	Type	Default Value
	float32	0.0
	Description: A value between 0.0 and 1.0 will adjust the start position of the soundfile. The sound needs to be fully buffered for this to function, ensure the buffer setting is set to 'True' for the sound in the Sound Repository.	
Begin Position	Type	Default Value
	float32	1.0
	Description: The current begin position offset as a fraction of the selected files total length, i.e. a value of 0.5 starts the file half way into normal play section.	
End Offset	Type	Default Value
	float32	0.0
	Description: A value between 0.0 and 1.0 will adjust the end position of the soundfile. The position is calculated relative to the end of the file (i.e. a value of 0.3 will result in an end position of 70% of the entire length of the file.) If the end offset is less than the start offset (after the BeginOffset calculation has been made, the end offset will result in a value of 0.0). The sound needs to be fully buffered for this to function, ensure the buffer setting is set to 'True' for the sound in the Sound Repository.	
End Position	Type	Default Value
	float32	1.0
	Description: The current end position offset as a fraction of the selected files total length, i.e. a value of 0.1 stops the file at 90% of it's normal end point.	

Control Inputs		
LibraryID	Type	Default Value
	playsound_library	0
	Description: Selects the library that contains the group and sound file.	
OutGain	Type	Default Value
	float32	1.0
	Description: Amplitude gain of file replay source. If the gain connection is blank then the gain scale factor is used as the gain value; otherwise the gain is the scale factor times the output result of the control object. Modifier: Multiply Modifier_default: 0.0 Range: 0.0 - Inf	
Pause	Type	Default Value
	boolean	FALSE
	Description: The pause state, a value of on freezes the soundfile playing, a value of off allows the play to continue from the current file position. If no external variable is connected to Pause, the modifier is used as the local value. Modifier: XOR Modifier_default: False	

Control Inputs		
Randomize	Type	Default Value
	boolean	FALSE
	Description: When True, the Playsound signal will have a counter running and will start the sound at the counter position when the sound is triggered, resulting in a random start position. Note that in the Sound Library the sound file buffer must be set to True in order for Randomize to have any effect.	

Audio Outputs		
Out	Type	Default Value
	boolean	False
	Description: Out is the output signal from the playsound component, which may be connected to another component.	

Control Outputs		
Playing	Type	Default Value
	boolean	False
	Description: The value is True if the component is actively playing audio.	

2.18. Pulse

Summary: This signal source produces a pulse stream signal which can be mixed in any proportion on any of the analog output channels.

Note: A pulse signal is similar to the square wave except it is limited to positive amplitudes.

Description: This signal source produces a pulse stream signal which can be mixed in any proportion on any of the analog output channels. Both gain, frequency and mark/space ratio can be controlled by input variables from elsewhere in the model, or from the host interface. Both the frequency and pulse amplitude can be modulated by other signals within the model while the pulse signal controls the depth of modulation.

The frequency of the output pulse signal is determined by a combination of the frequency variable and the frequency modulating signal. If no signal source is connected to the frequency modulating signal field, only the frequency variable is used.

If a signal source is connected to the frequency modulating signal field, the actual frequency of the waveform is varied according to the amplitude of the modulating signal. The degree to which the frequency varies is controlled by the modulation depth. The calculation is:

$$\text{ActualFreq} = \text{Freq} \times (1 + (\text{ModDepth} \times \text{ModSignal}))$$

where Freq is the frequency variable and ModSignal is the amplitude of the modulating signal. A modulating signal with an active state of Off should be treated as an amplitude of 0.0.

When the waveform frequency is less than or equal to 0, the wave component does not generate a signal.

The amplitude of the output pulse signal is determined by a combination of the gain variable and the amplitude modulating signal. If no signal source is connected to the amplitude modulating signal field, only the gain variable is used.

If a signal source is connected to the amplitude modulating signal field, the actual amplitude of the output pulse is varied according to the amplitude of the modulating signal. The degree to which the amplitude varies is controlled by the modulation depth. The calculation is:

$$\text{ActualAmp} = \text{Gain} \times (1 + (\text{ModDepth} \times \text{ModSignal}))$$

where Gain is the gain variable and ModSignal is the amplitude of the modulating signal. A modulating signal with an active state of Off should be treated as an amplitude of 0.0.

When the waveform amplitude is less than or equal to 0, the wave component does not generate a signal.

Audio Inputs		
AmpMod-Signal	Type	Default Value
	audio	n/a
Description: Amplitude modulation signal, controls the amplitude of the final output pulse. To avoid unpredictable behavior care should be taken to ensure that the product of modulation depth and modulation signal does not span a range greater than -1.0 to +1.0.		

Control Inputs		
AmpMod-Depth	Type	Default Value
	float32	1.0
Description: Amplitude modulation depth value, controls the effect of the frequency modulation signal. Usually it falls in the range 0 to 1.0, when used in conjunction with a unity gain modulation signal. If no external variable is connected to AmpModDepth, the value of the scaler is used as the frequency modulation depth. To avoid unpredictable behavior care should be taken to ensure that the product of modulation depth and modulation signal does not span a range greater than -1.0 to + 1.0. Modifier: Multiply Modifier_default: 0.0 Range: 0.0-1.0		

Control Inputs		
Frequency	Type	Default Value
	float32	1.0
	Description: Frequency (in Hertz) of wave generated by the waveform synthesizer. The frequency is the number of oscillations made per second for the given waveform. If no external variable is connected to Frequency, the value of the scaler is used. Modifier: Multiply Modifier_default: 0.0 Range: 0.0 - 1/2*Sample Rate	
FreqMod-Depth	Type	Default Value
	float32	1.0
	Description: Frequency modulation depth value, controls the effect of the frequency modulation signal. If no external variable is connected to FreqModDepth, the value of the scaler issued is the frequency modulation depth. To avoid unpredictable behavior care should be taken to ensure that the product of modulation depth and modulation signal does not span a range greater than -1.0 to +1.0. Modifier: Multiply Modifier_default: 0.0 Range: 0.0-1.0	
FreqMod-Signal	Type	Default Value
	boolean	False
	Description: Frequency modulation signal, controls the actual generated frequency. To avoid unpredictable behavior care should be taken to ensure that the product of modulation depth and modulation signal does not span a range greater than -1.0 to +1.0.	

Control Inputs		
Gain	Type	Default Value
	float32	1.0
	Description: Amplitude gain of the waveform. If no external variable is connected to Gain, the value of the scaler is used. Modifier: Multiply Modifier_default: 0.0 Range: 0.0 - Inf	
Mark-Space Ratio	Type	Default Value
	float32	1.0
	Description: The pulse width or mark to space ratio of the wave. The pulse is positive relative to the period of the wave. (e.g. a value of 0.5 means in a given period, the pulse output will be +Gain for half of the time and zero for the other half). If no external variable is connected to MarkSpace, the value of the scaler is used. Modifier: Multiply Modifier_default: 0.5 Range: 0.0 - 1.0	

Audio Outputs		
OutSignal	Type	Default Value
	audio	n/a
	Description: OutSignal is the output signal from the pulse component, which may be connected to another component or directed to an output highway.	

2.19. Pulse Sequence

Summary: The Pulse Sequence component can generate a repeating series of up to eight pulses of arbitrary pulse width, pulse amplitude, and timing.

Description: The Pulse Sequence is a signal component that can generate a repeating series of up to eight pulses of arbitrary pulse width, pulse amplitude, and timing. Typically, this signal is used to frequency or amplitude modulate other signals.

Part (a) of the figure shows what the various parameters in the Pulse Sequence specify. The paint count specifies the number of pulses, while the delays, amplitudes, and durations are specified as shown.

The pulse sweep time can be modulated by an external signal. How the pulses within the sweep act depends on whether the paint times are defined as “Fixed” or “Fractional”.

For fixed paint times, the paint times and durations are defined in terms of the number of microseconds after the initiation of the sweep. Modulating the sweep time will not affect the time of the pulses, and if the sweep time cuts off a pulse it will not be generated. Part (b) of the figure shows the same pulse sequence as part (a), but with the sweep time shortened in fixed paint time mode.

For fractional paint times, the paint times and durations are defined in terms of the fraction of the total sweep time. Modulating the sweep time will compress (or extend) the pulses and move them closer together (or farther apart). Part (c) of the figure shows the same pulse sequence as part (a), but with the sweep time shortened in fractional paint time mode.

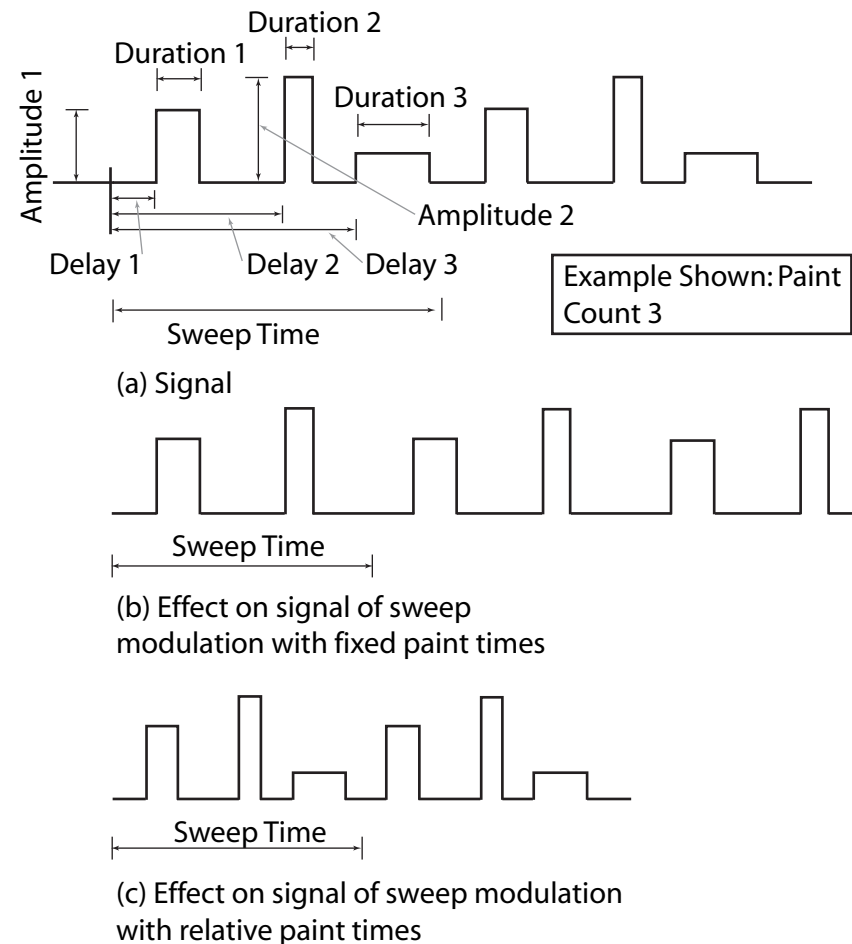


Figure 5: Pulse Sequence

Audio Input		
GainMod	Type	Default Value
	audio	n/a
	Description: Provides a connection to a signal which will modulate the amplitude of the pulse sequence.	
Sweep Mod	Type	Default Value
	audio	n/a
	Description: Provides the audio input connection for the modulation of the sweep time.	

Control Input		
Durations	Type	Default Value
	float32	1.0
	Description: Provide the width of each pulse, in microseconds if in Fixed mode, or as a fraction of the sweep time in Fractional mode. Note: The resolution of the pulse durations is 1/sample rate. For a sample rate of 48kHz, the resolution is 20.8333 microseconds.	
Gain	Type	Default Value
	float32	1.0
	Description: Amplitude gain of pulse. If the gain connection is blank then the gain scale factor is used as the gain value, else the gain is the scale factor times the output result of the control object.	

Control Input		
GainMod En	Type	Default Value
	boolean	false
	Description: Control parameter to enable connection to provide amplitude gain control from elsewhere in model.	
Paint Count	Type	Default Value
	uint8	0
	Description: The paint count is the number of pulses in a sweep. Allowable values are from 1 to 8. A value of 0 disables the pulse sequence.	
Sweep-ModEn	Type	Default Value
	boolean	false
	Description: Control parameter to enable connection to provide sweep modulation control from elsewhere in model.	
Sweep Mod Depth	Type	Default Value
	float32	0.0
	Description: Provides the modulation depth for the sweep modulation signal. This should be between 0 and 1. A one means that if the modulation signal has a gain of 1, then the sweep time (or frequency) will get modulated between zero and twice it's normal value. A modulation depth of zero means that no sweep modulation will occur.	

Control Input		
Sweep Time	Type	Default Value
	float32	0.0
	Description: Provides the length of the sweep that the pulses are contained in, in microseconds. See the figure above for details. Note 1: If the PaintTimes flag is set to fractional, this field will be a frequency in hertz. Note 2: The timing resolution of the sweep time is 1/sample rate. For a sample rate of 8kHz, the resolution is 125 microseconds.	

Internal Parameters		
Amplitude	Type	Default Value
	float32	0.0
	Description: Provides the height of each pulse. Each pulse height should be between 0 and 1.	
Delays	Type	Default Value
	float32	0.0
	Description: Provide the delay, for each pulse, between the beginning of the sweep and the beginning of each pulse. If a pulse delay puts a pulse outside the sweep, that pulse will not be generated. Note: The resolution of the Delays is 1/sample rate. For a sample rate of 8kHz, the resolution is 125 microseconds.	

Internal Parameters		
Paint Times	Type	Default Value
	pulse_step_mode	Fixed
	Description: This is a flag which is set either to Fixed or Fractional. If set to Fixed, the sweep time, delay times, and durations are all measured in microseconds. If set to Fractional, the sweep is a frequency in hertz, and the Durations and delays are specified as a fraction of the sweep time.	

Audio Outputs		
OutSignal	Type	Default Value
	audio	n/a
	Description: OutSignal is the output signal from the component, which may be connected to another component or directed to an output highway.	

2.20. PulseStep

Summary: The Pulse Step can generate a repeating series of up to sixteen sequential pulses.

Description: The Pulse Step can generate a repeating series of up to sixteen sequential pulses of arbitrary pulse width and pulse amplitude. The pulses follow immediately one after the other - there is no space between them. Typically, this signal is used to frequency or amplitude modulate other signals.

The string of pulses can be looped repeatedly with a specified loop time, or can be triggered as a "one-shot" sequence. When looped, the loop time can be modulated with another signal. If the loop time is modulated, the pulses can be shortened and lengthened with the loop time (fractional step times) or kept at a constant length (fixed step times.)

At the end of the pulses, the output of the signal goes to a constant amplitude which is specified in the "amplitude off" field (i.e. the amplitude when all the pulses are off).

In addition, the gain of the signal can be modulated by an external signal.

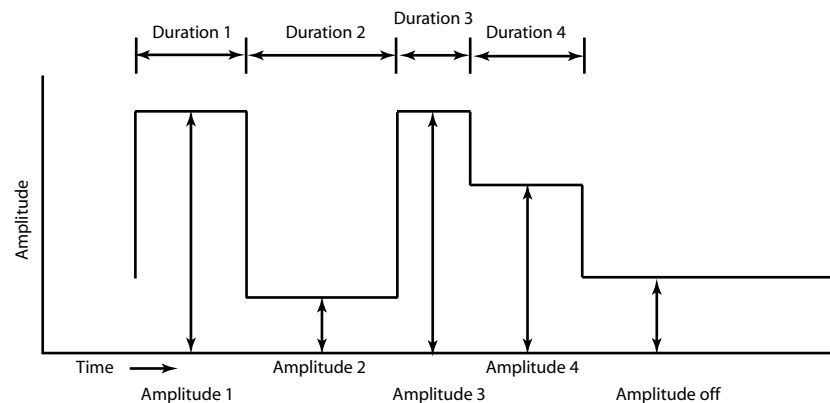


Figure 6: Pulse Step Signal

Audio Input		
GainMod	Type	Default Value
	audio	n/a
	Description: Provides a connection to a signal which will modulate the amplitude of the pulse sequence.	
LoopMod	Type	Default Value
	audio	n/a
	Description: Provides a connection to a control component which gives the loop modulation depth.	

Control Inputs		
Amplitudes 1-16	Type	Default Value
	float32	1.0
	Description: Provides the amplitudes for the pulses. See the figure in the Pulse Step description.	
Durations 1-16	Type	Default Value
	float32	1.0
	Description: Provides the durations for the pulses in microseconds (fixed times) or as a fraction of the sweep time (fractional times). See the figure in the Pulse Step description.	
Gain	Type	Default Value
	float32	1.0
	Description: Amplitude gain of the signal. If the gain connection is blank then the gain scale factor is used as the gain value, or the gain is the scale factor times the output result of the control component.	

Control Inputs		
Gain-ModEn	Type	Default Value
	boolean	false
	Description: Enables the gain modulation.	
LoopMod-Depth	Type	Default Value
	float32	1.0
	Description: Provides the modulation depth for the loop modulation signal. This should be between 0 and 1. A one means that if the modulation signal has a gain of one, then the loop time (or frequency) will get modulated between zero and twice it's normal value. A modulation depth of zero means that no loop modulation will occur.	
Loop-ModEn	Type	Default Value
	boolean	false
	Description: Enables the loop modulation.	
LoopTime (or frequency)	Type	Default Value
	float32	1.0
	Description: The Loop Time (fixed mode) or loop frequency (Hz) (fractional mode) determines the rate at which the pulses repeat themselves. If the loop time is zero, the Pulse Step component goes into one-shot mode. In one-shot mode, the pulses are generated only when triggered by the Step Count changing from 0 to a non-zero value. The rest of the time the output is given by the Amplitude Off value.	

Control Inputs		
StepCount	Type	Default Value
	uint8	0
	Description: This number is an integer which provides the number of pulses. It must be between 1 and 16. Setting the value to zero turns the pulse stream off, so the signal output is just the “amplitude off” value. If the pulse step signal is in one-shot mode, then changing the Step Count from zero to a non-zero value will trigger the pulse stream to start.	

Audio Outputs		
OutSignal	Type	Default Value
	audio	n/a
	Description: OutSignal is the output signal from the component, which may be connected to another component or directed to an output highway.	

Internal Parameters		
AmplitudeOff	Type	Default Value
	float32	0.0
	Description: Provides the amplitude of the signal when no pulses are being generated. This occurs either because the Step Count is zero or in the “dead time” at the end of a loop when there are no more pulses.	

Internal Parameters		
PlayAll	Type	Default Value
	boolean	False
	Description: This parameter is used in one-shot mode. In normal mode, if the step count is toggled from 0 to a non-zero value and back to zero, the pulses will stop playing immediately when the Step Count goes back to zero. In Play All mode, the pulses will finish playing to the end of the sequence.	
Step Mode	Type	Default Value
	pulse_step_mode	fixed
	Description: This parameter determines whether the pulse durations are given in absolute times in microseconds (fixed mode) or as a fraction of the loop time (fractional mode). When the loop time is modulated, fractional mode causes the pulses to shorten and lengthen in proportion to the loop time. In fixed mode, the pulses remain at their fixed durations when the loop time is varied. If the loop time becomes less than the sum of the pulse durations in fixed mode, the end pulses will get cut off.	

2.21. PulseStream

Summary: The Pulse Stream signal source is a sophisticated signal source which generates a stream of pulses. Like the Pulse signal, the pulses have an amplitude between 0 and 1.

Description: The pulse stream is typically used to frequency or amplitude modulate other signals, to provide time varying tones. In the Pulse Stream Signal figure shown below the pulse width and PRI (Pulse Repetition Interval) are shown. The PRI is usually the Main PRI modulated by some other signal or function. There are several ways to modulate the timing between pulses. These different modulation methods are called Pulse types, and are specified in the Pulse Type field. Each Pulse type has a number from 1 to 255 and a name. For each pulse type, some of the parameters are ignored. See the PulseType table to see which pulse types use which parameters.

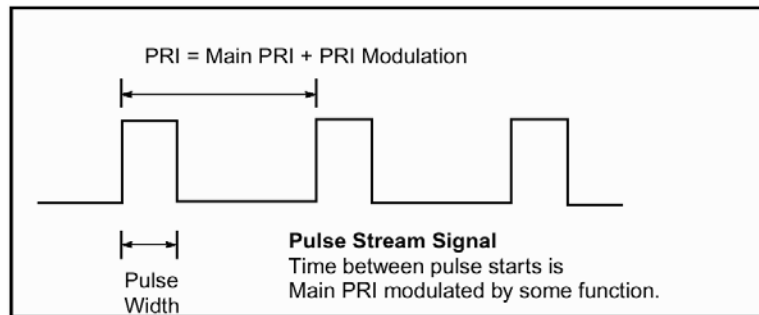


Figure 7: Pulse Stream Signal

The Signal Modulated figure below shows how the first few Pulse Types work. The graph shows the PRI (Pulse Repetition Interval) as a function of time. The specific example is a triangle modulated pulse stream (Pulse type 3, Triangle.) Built into the object are sine, triangle, sawtooth and square wave modulations. You can also use an arbitrary signal to modulate the PRI by selecting Pulse Type 9 (external) and entering a signal in the PRI Modulation field. For the built-in modulation signals, you need to specify a modulation frequency and modulation depth. For the external pulse type, the modulation frequency is ignored and the modulation depth becomes a scale factor for the external modulation signal.

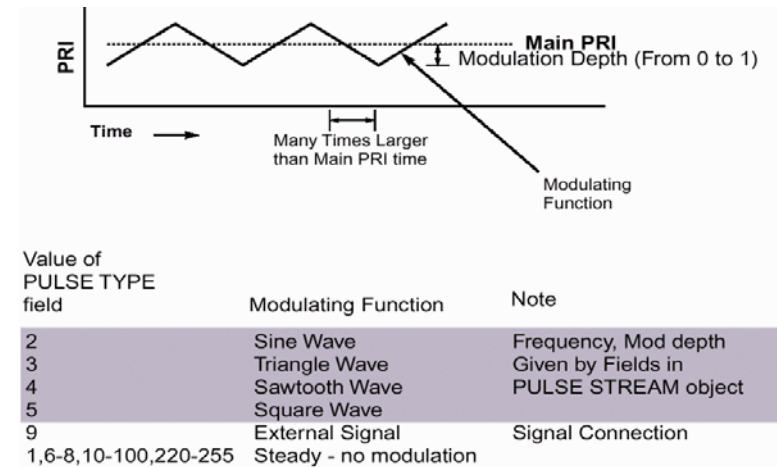


Figure 8: Signal Modulated

Pulse types 101 to 150 are called dwell1 to dwell 50. The spacing between pulses (PRI) is modulated in a step like fashion. The length of time on each step is given by the dwell time, and the number of steps is the dwell number plus one. (e.g., for dwell5 there will be six steps.) The steps are evenly spaced, and the modulation depth, as shown, gives their height.

The PRI time varies between

$$\text{PRI Main} \times (1 - \text{Modulation Depth})$$

and

$$\text{PRI Main} \times (1 + \text{Modulation Depth})$$

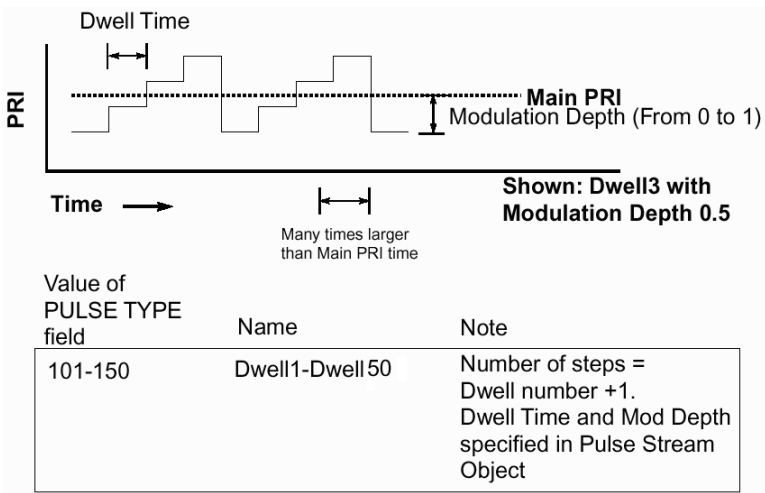


Figure 9: Dwell

The figure below shows the Random Dwell Pulse Type (Pulse types 151-199). The random dwell is the same as the dwell, except that instead of stepping sequentially through the levels it jumps randomly among them. The number of levels is given by the Random Dwell number + 1 - i.e., Random Dwell 5 would jump through 6 different PRI modulation times.

In addition, there is a Random Dwell with no number (Pulse type 199). With this pulse type, the PRI time jumps randomly throughout it's allowed range, staying at each PRI time for a dwell time.

The allowed range for the random dwell is between

$$\text{PRI Main} \times (1 - \text{Modulation Depth})$$

and

$$\text{PRI Main} \times (1 + \text{Modulation Depth})$$

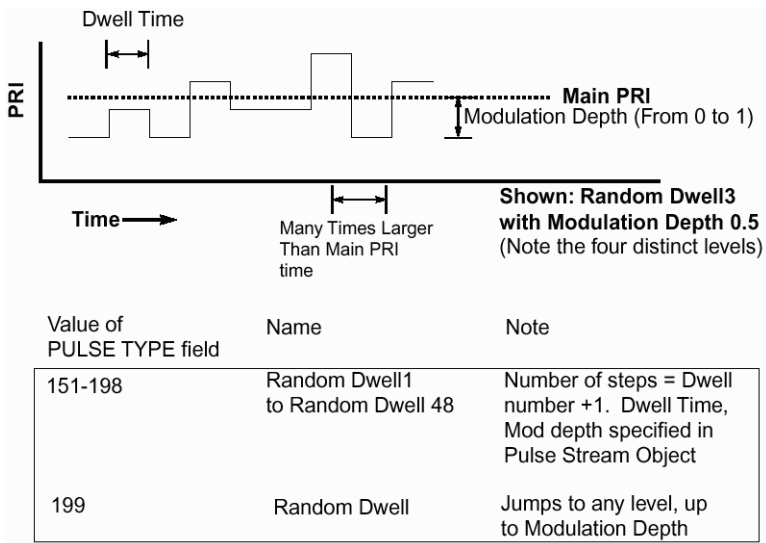


Figure 10: Random Dwell

The Stagger Pulse types operate differently from the other pulse types mentioned before. Instead of modulating the PRI, up to eight PRI values are given which define the spacing of the pulses. The number of pulses in the stream is given by the stagger number (e.g. Stagger 4 has four pulses per cycle).

The length from cycle to cycle is given by the Main PRI value, while the placement of the intermediate pulses is shown in the diagram below.

The stagger pulse type is the only one to use the Stagger PRI values. It ignores all of the PRI modulation fields.

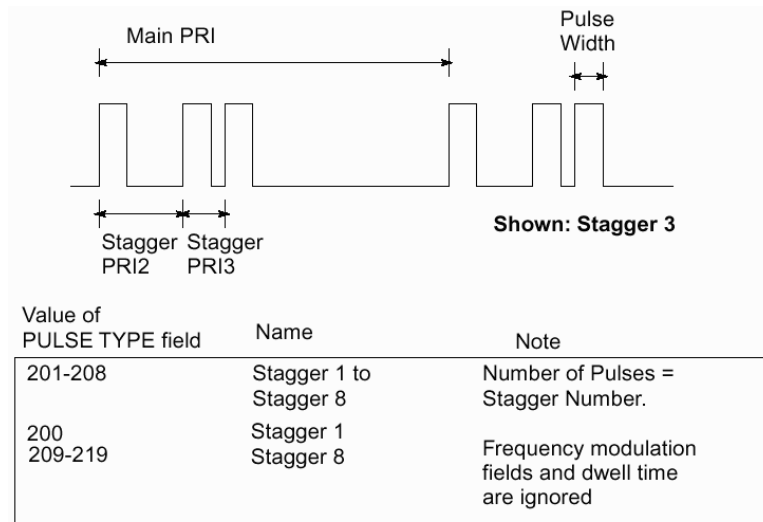


Figure 11: Stagger

This pulse stream type table summarizes which parameters are used by which pulse types.

Pulse Type Number	PulseType Name	Stagger PRIs	PRI Freq	Modulation Depth	Dwell Time
1	steady	N	N	N	N
2	sine	N	Y	Y	N
3	triangle	N	Y	Y	N
4	sawtooth	N	Y	Y	N
5	square	N	Y	Y	N
6-8	steady	N	N	N	N
9	external	N	N	Y	N
10-100	steady	N	N	N	N
101-150	dwell1-dwell50	N	N	Y	Y
151-198	random dwell1 - random dwell 48	N	N	Y	Y
199	random dwell	N	N	Y	Y
200	stagger1	Y	N	N	N
201-208	stagger1-stagger8	Y	N	N	N
209-219	stagger8	Y	N	N	N
220-255	steady	N	N	N	N

Audio Inputs		
GainMod	Type	Default Value
	audio	n/a
	Description: Connection to a signal which will modulate the amplitude of the pulse stream.	
PRI_Mod	Type	Default Value
	audio	n/a
	Description: Provides a connection to a signal which modulates the Main PRI (Pulse Repetition Interval) value. This field is only used by Pulse Type 9 (external).	

Control Inputs		
DwellTime	Type	Default Value
	float32	1.0
	Description: This is used by the dwell and random dwell pulse types (numbers 101 to 199). It gives the value, in seconds, that the PRI stays on a particular value. See the description above for details.	
Gain	Type	Default Value
	float32	1.0
	Description: The amplitude gain of the pulse stream signal.	
GainMod Enable	Type	Default Value
	Boolean	False
	Description: Determines whether amplitude gain modulation is enabled. Value is True if a signal is connected to GainMod.	

Control Inputs		
Main_PRI	Type	Default Value
	Float32	1.0
	Description: Provides the basic spacing between pulses, as measured from the beginning of successive pulses. This spacing can be modulated by the different pulse types. The Main PRI is measured in microseconds.	
PRI_Mod Depth	Type	Default Value
	Float32	1.0
	Description: A number between 0 and 1 which determines the modulation depth for the Main PRI modulation. A 0 indicates no modulation. The range of PRI values will be from $PRI_{Main} \times (1 - \text{Modulation Depth})$ to $PRI_{Main} \times (1 + \text{Modulation Depth})$ This field is ignored for the Steady and Stagger Pulse types.	
PRI_Mod Freq	Type	Default Value
	Float32	1.0
	Description: For pulse types 2 through 5 (sine, triangle, sawtooth, square), this field provides the frequency of the signal modulating the Main PRI. It is ignored for other pulse types.	
PulseType	Type	Default Value
	uint8	0
	Description: This field determines the type of pulse stream. The pulse type is a number between 0 and 255. Each number has an associated name which appears next to the number. A pulse type of zero turns off the pulse stream.	

Control Inputs		
Pulse-Width	Type	Default Value
	float32	1.0
	Description: Provides the width of the pulses in the pulse stream, in microseconds.	
Stagger_P RI_2 - Stagger_P RI_8	Type	Default Value
	Float32	1.0
	Description: Gives the stagger time, in microseconds, for the stagger pulse types. These values are only used by the stagger pulse types (pulse types 200 to 219). See description above for a detailed explanation.	

Audio Output		
OutSignal	Type	Default Value
	audio	n/a
	Description: OutSignal is the output signal from the PulseStream component, which may be connected to another component or directed to an output highway.	

Internal Parameter		
PulseType-Name	Type	Default Value
	string	n/a
	Description: Displays the currently selected pulse type.	

2.22. Record Replay

Summary: The Record Replay component provides the ability to record and replay digitally encoded soundfiles (known simply as “recordings” in ACE vernacular). The component is capable of recording any signal source within a model and features host control inputs for controlling record-replay modes, file selection and file position, gain and other parameters.

Description: The Record Replay component provides the ability to record any input signal and to replay from a pre-recording file (which may already exist) on the hard disk. Host inputs are provided to control record-replay-stop operation, file and group index selection, file pointer repositioning during record and replay, and gain adjustment. The Record Replay component features a host-controlled selectable link to the recording index, enabling dynamic selection of specific files inside specific groups during record and replay.

Additional Record replay file information:

- I. Record replay files (called recordings) are processed and stored separately from playsound files (called sounds).
- II. Record Replay component cannot replay sounds and the Playsound component cannot replay recordings.
- III. The internal recording file format TSR is a raw uncompressed audio file. See the Remote Management System > Archive Recordings to convert the file into a wave (.wav) file.
- IV. Unlike playsound files, record replay files do not contain meta data (internal file settings for start-stop positions, playall-normal).
- V. An import-export utility is required for file conversion to a wave file.
- VI. TSR files are stored on the Target hard drive in the following location:

```
/var/local/ast/recordreplay
```

The files have the following name structure:

```
library000_groupN_indexM.tsr
```

where ‘N’ is the group number defined in the component and ‘M’ is the index number defined in the component.

Audio Inputs		
SignalIn	Type	Default Value
	audio	n/a
	Description: Audio linked in to this field will be recorded. When in replay mode this is unused. Audio level is controlled by the gain field.	

Control Inputs		
Command	Type	Default Value
	uint8	0
	Description: Three states exist in the Record Replay component. They are: 0 = Stopped 1 = Replay 2 = Record The command variable should always transition through the stopped state when switching from record to replay or vice-versa.	
Gain	Type	Default Value
	float32	1
	Description: Amplitude gain control for the record or replay audio. This effects both signal in and signal out.	

Control Inputs		
Position	Type	Default Value
	int32	0
Description: The position field is used when the command field leaves the stop state and goes into the record or replay mode. In other words, this value is applicable when you transition the command variable from 0 to 1 or 0 to 2. The meaning of the field will depend on the loop setting being TRUE/FALSE and also on the rate setting in kHz. The units are in milliseconds. Loop = False Position 0: the beginning of the file if you haven't recorded, or where you left off in the file if you have recorded. Position 2: 48kHz at the beginning of the file Position 8: 16kHz at the beginning of the file Position 16: 8kHz at the beginning of the file Loop = TRUE When in continuous loop mode, a positive value provides an offset forward in the file, while a negative offset repositions the file back from the last stop position. When not in loop mode, a positive position provides an offset from the start of the file, while a negative number is an offset from the end of the file. A value of zero leaves file pointer at its current position.		

Control Inputs		
Reset	Type	Default Value
	boolean	False
Description: The Reset control is used to reload the file resource currently used. The Reset function will only function if the component is currently in the "Stopped" mode (i.e., Command is set to 0). Furthermore, the Reset function is only triggered on the rising edge of the signal, when it changes from False to True. An example use case for this control is if the user intends to swap or archive record/replay files on the server at runtime. The workflow would be 1) record a segment, 2) stop recording, 3) move the record files off the server, such as for archiving, 4) trigger the Reset control, and 5) start recording again. With this workflow a new record file will be created on the fifth step.		
Sound Index	Type	Default Value
	uint16	0
Description: Set the index number for the recording. Every recording must have a unique group and index. The value must be non-zero for replay.		

Audio Outputs		
SignalOut	Type	Default Value
	audio	n/a
	Description: Output signal of the Record Replay component. Audio level is controlled by the gain variable and read from the raw audio record/replay file. Audio is only output when in replay mode.	

Control Outputs		
Current Position	Type	Default Value
	int32	0
	Description: The current position displays the position of the file in milliseconds. This control is updated as the component records or replays audio. If the component stops the audio, the position will hold at its current value until replay or recording restarts.	
Current FileLength	Type	Default Value
	int32	0
	Description: After a recording this displays the current file length in milliseconds of the selected record/replay file.	
FileMode	Type	Default Value
	player_state	STOPPED
	Description: Displays the stopped recording or playing state of the file. Displays “RECORDING” when the file is recording and displays “REPLAY” when file is replaying.	
MaxFileLength	Type	Default Value
	int32	0
	Description: Displays the maximum file length in milliseconds. MaxFileLength is based on the length setting.	

Control Outputs		
New Position	Type	Default Value
	int32	0
	Description: The point in the file where it will resume if it begins recording or replaying.	
Position Offset	Type	Default Value
	int32	0
	Description: The Position Offset matches the position input and displays in milliseconds.	

Internal Parameters		
GroupID	Type	Default Value
	uint16	0
	Description: Use to identify the record/replay file group. Each file is organized as part of a specific group and may be stored on the hard disk. Group ID is used in conjunction with the index to point to a unique file.	
Length	Type	Default Value
	uint32	0
	Description: Maximum duration of a non-looped recording. Loop duration of a recording when in loop mode. Units are in seconds.	
Loop	Type	Default Value
	boolean	False
	Description: Sets the component to continuous play in loop. If false, the component will stop recording or playing after reaching the end of the file. If loop is set to true, when the replay object is at the end of the file, it will continue to record or replay starting at position 0.	
Rate	Type	Default Value
	rate_hz	rate_8kHz
	Description: Record audio rate selection. Configuration options are 8kHz, 16kHz, and 48kHz. Note: Rate change requires a model reload.	

2.23. SimpleMixer

Summary: The SimpleMixer provides quick and easy mixing of up to 32 input signals.

Description: This is a no frills mixer intended to be the absolute fastest way to mix audio. Up to 32 signals can be linked to the InSginals connection and they are then mixed together equally. One overall OutGain is provided if the level needs to be moved up or down.

The SimpleMixer is most useful for cases when the CPU processing is at a premium, or when individual audio mixing controls are not necessary.

Audio Input		
InSignals	Type	Default Value
	audio	n/a
	Description: An input connection for up to 32 audio signals.	

Control Input		
OutGain	Type	Default Value
	float32	1.0
	Description: A volume control for the overall level of the output mix.	

Control Output		
OutSignal	Type	Default Value
	audio	n/a
	Description: The audio output of the component, containing an equal amount of each input.	

2.24. Sequencer

Summary: The Sequencer plays sound files in a specified order.

Description: The Sequencer cycles through the indexes (soundfiles) when the trigger is set to true. The Sequencer basically drives the Playsound component.

Audio Output		
OutSignal	Type	Default Value
	float32	1.0
	Description: OutSignal is the output signal from the Sequencer component, which may be connected to another component or directed to an output highway.	

Control Inputs		
Continu-ous	Type	Default Value
	boolean	true
	Description: The Continuous Mode flag determines the Sequencer behavior when the Trigger is held TRUE. If this flag is TRUE, the Sequencer continually steps through the sequence. When it reaches the End Value, it returns to the Start Value and begins the sequence again. If this flag is FALSE, the Sequencer does only one pass through the sequence. It stops sequencing when it reaches the End Value.	
Delay	Type	Default Value
	uint32	1
	Description: The length of time (in seconds) between sequencing cycles when the Sequencer is in Continuous Mode.	

Control Inputs		
Playall	Type	Default Value
	boolean	true
	Description: When trigger is turned off playall determines if the sequencer stops immediately or plays to the end of the file. Defaults to true which plays to end of file.	
Playing	Type	Default Value
	boolean	true
	Description: Attaches playsound files to the component.	
Reset	Type	Default Value
	boolean	true
	Description: If reset flag is set the Sequence is set to 0.0.	
Trigger	Type	Default Value
	boolean	true
	Description: A Trigger state of TRUE initiates sequencing. The Continuous Mode flag determines how the sequencer behaves when the Trigger is set to False.	

Control Outputs		
GroupID	Type	Default Value
	playsound_group	0
	Description: The value of the GroupID. The GroupID is used to select a group from within a library.	
SoundIdx	Type	Default Value
	playsound_sound	0
	Description: The value of the file index to be played.	
Trigger-Out	Type	Default Value
	boolean	false
	Description: When set to true trigger initiates the sequence.	

2.25. Volume Control

Summary: Provides a host control for overall volume level in the model.

Description: Sets the main volume level. When Mute is set to True the volume level is 0.

Control Inputs		
Mute	Type	Default Value
	boolean	false
	Description: When mute is true it sets the volume level to 0.	
Volume	Type	Default Value
	float32	1.0
	Description: Sets the main volume, overall controlled by the host.	

Control Outputs		
Volume Level	Type	Default Value
	float32	1.0
	Description: Resulting output of the volume setting.	

2.26. Vox

Summary: The Vox component allows voice activated or push-to-talk (PTT) control over an audio input signal. An optional filter may be applied to the input signal. If the filtered input signal level exceeds the Vox threshold level, the Vox component will output the signal. Once the input signal level has dropped below the threshold, the Vox component will remain active for the period of time specified by the Vox delay. Alternately, the Vox component can operate in PTT mode in which the Vox component outputs a signal only when the PTT control is set.

Description: The Vox component operates on an input signal. The source of the input signal is an external component that is connected to the Vox via an InSignal variable.

The Vox component first applies a filter to the input signal. The output from the filter is active only if the input signal source is active.

The filter is controlled by the parameters: Type, Filter Frequency, and Q Factor. Type determines the type of filter applied to the input signal (Off, Lowpass, Highpass, or Bandpass). If Type is set to Off, no filtering should be applied to the input signal. Filter Frequency (in Hertz) provides the rolloff frequency of the filter. The meaning of the frequency depends on the filter type as follows:

Filter Type	Frequency Meaning
Off	Ignored
Lowpass	Upper bound frequency
Highpass	Lower bound frequency
Bandpass	Center frequency with bandwidth controlled by Q Factor (see below)

Q Factor is the filter quality factor, which effectively determines the filter rolloff for lowpass and highpass filters and the filter bandwidth for bandpass filters. Q Factor is inversely proportional to the filter rolloff or bandwidth.

The filter parameters are used to derive the filter coefficients for a 2-pole IIR filter. If Q Factor is 0 or less, a very low non-zero value should be used for the k calculation. If the Filter Type is Off, the filter output signal is the input signal.

If the output of the filter is active, the Vox component applies Vox or PTT control to the filtered signal. The Vox component includes a Vox enable flag and PTT control. The Vox and PTT controls are specified by an externally controlled variable and an XOR logic gate control. XOR logic is applied to the external variable value and the gate control to derive a final value for the Vox enable and PTT flags. If no external variable is connected, the value of the logic gate control becomes the input value.

These two boolean inputs control when the Vox outputs audio as follows:

I. Vox Enable = False, PTT = False - No Output

II. Vox Enable = True, PTT = False - Output active only when signal level exceeds the threshold

III. Vox Enable = False, PTT = True - Output active always (regardless of signal level)

IV. Vox Enable = True, PTT = True - Output active always (regardless of signal level)

For case II, if the signal level drops below threshold, the Vox component will continue to output a signal for the duration specified in the Vox delay parameter. The threshold level is specified by the Vox level control, which consists of a connection to an external control variable and an offset. The value of the external variable is added to the offset to derive the Vox threshold level. If no external variable is connected, the value for the offset becomes the Vox threshold level.

An amplitude gain control is applied to the Vox output. This output gain control consists of a connection to an external controlled variable and a scale factor. The value of the external variable is multiplied by the scale factor to derive the output gain. If no external variable is connected, the value for the scale factor becomes the signal gain.

The Vox component output audio should only be active if both of the following conditions are true:

I. The output from the filter is active.

II. The derived output gain (external variable * scale factor) is greater than 0.0.

The Summary View section contains two variables which are not an input, output, or internal parameter. These are Filter Output Signal Level and Output Signal Level. Filter output Signal Level reflects the RMS signal power at the output of the Vox filter. This variable will be non-zero whenever the Input Signal is active. Output Signal Level is the RMS signal power of the output from the Vox component. This variable will only be non-zero if the component output is active. It is possible for the Filter Output Signal Level to be non-zero while Output Signal Level is zero. This would happen if the filtered signal does not exceed the Vox threshold. If the output signal is active, both variables would have the same value.

Audio Inputs		
InSignal	Type	Default Value
	audio	n/a
	Description: InSignal is the input signal into the Vox component. The filter and the Vox logic will be applied to this signal to determine the output signal.	

Control Inputs		
OutGain	Type	Default Value
	float32	1.0
	Description: OutGain applies amplitude gain control to the primary output signal. If no external control is connected to OutGain, the scale factor is used as the OutGain value. Modifier: Multiply Modifier_default: 1.0	
PTT	Type	Default Value
	boolean	False
	Description: In PTT mode, the input signal is output only when PTT is set. When PTT is set in Vox mode, the Vox component behaves as if in Hot Mic mode (i.e. it will always output the signal). If no external control is connected to PTT, the XOR modifier is used as the PTT value. Modifier: XOR Modifier_default: FALSE	

Control Inputs		
VoxEnable	Type	Default Value
	boolean	False
	Description: VoxEnable is the control flag for Vox mode. If this variable is set to false, the Vox component is configured for PTT mode. If no external control is connected to VoxEnable, the XOR modifier is used as the VoxEnable value. Modifier: XOR Modifier_default: TRUE	
VoxLevel	Type	Default Value
	float32	0.0
	Description: VoxLevel provides the Vox threshold level. When Vox mode is enabled, the input signal level is compared to VoxLevel and fed through only if it exceeds it. If no external control is connected to VoxLevel, the offset is used as the VoxLevel value. Modifier: ADD Modifier_default: 0.002	

Audio Outputs		
OutSignal	Type	Default Value
	audio	n/a
	Description: Outsignal is the output signal from the Vox component.	

Control Outputs		
FilterSignalLevel	Type	Default Value
	filter32	n/a
	Description: Indicates the sound level of the filtered signal.	
OutSignalLevel	Type	Default Value
	filter32	n/a
	Description: The same as the filtered signal level except when the Vox is not outputting audio then the value is 0.	

Internal Parameters		
FilterFreq	Type	Default Value
	float32	2000.0
	Description: FilterFreq (in Hertz) provides the rolloff frequency of the filter. For lowpass filters, FilterFreq acts as an upper bound frequency. For highpass filters, FilterFreq acts as a lower bound frequency. For bandpass, FilterFreq acts as a center frequency and the filter bandwidth is determined by FilterQ. Range: 0.0 - Sample Rate / 2	
FilterQ	Type	Default Value
	float32	0.7071
	Description: FilterQ is the filter quality factor, which effectively determines the filter rolloff for lowpass and highpass filters and the filter bandwidth for bandpass cannot be less than or equal to 0.	

Internal Parameters		
FilterType	Type	Default Value
	filter_type2	LowPass
	Description: FilterType determines the type of filter applied to the input signal (Lowpass, Highpass, and Bandpass). This filtering occurs before the Vox compares the signal level to the Vox level. If not filtering is desired, FilterType should be set to Off. Range: Off, LowPass, BandPass, HighPass, LowPassQ, BandPassQ, HighPassQ, and Notch. The three Q filters are amplitude adjusted such that the filter has unity gain at the roll-off frequency, and maintains this gain as the quality factor is increased. The band-pass filters have the low-pass and high-pass poles at the same roll-off frequency.	

Internal Parameters		
VoxDelay	Type	Default Value
	float	2.0
	Description: VoxDelay is the amount of time (in sec.) after the input signal level falls below the Vox level that the Vox component will continue to output audio.	

2.27. Wave

Summary: This signal source produces a waveform signal, which can be mixed in any proportion on any of the analog output channels. The waveform type may be sine, sawtooth, triangle, or square.

Description: This signal source produces a waveform signal, which can be mixed in any proportion on any of the analog output channels. Both amplitude and frequency can be controlled by input variables from elsewhere in the model, or from the host interface. The frequency can also be modulated by another signal within the signal processor, with the model having control over the depth of modulation. The waveform signal type can be sine, sawtooth, triangle or square.

The waveform frequency is determined by a combination of the frequency variable and the frequency modulating signal. If no signal source is connected to the frequency modulating signal field, only the frequency variable is used.

If a signal source is connected to the frequency modulating signal field, the actual frequency of the waveform is varied according to the amplitude of the modulating signal. The degree to which the frequency varies is controlled by the modulation depth. The calculation is:

$$\text{ActualFreq} = \text{Freq} \times (1 + (\text{ModDepth} \times \text{ModSignal}))$$

where Freq is the Frequency variable and ModSignal is the amplitude of the modulating signal. A modulating signal with an active state of OFF should be treated as an amplitude of 0.0.

When the waveform frequency is less than or equal to 0, the wave component does not generate a signal.

The amplitude of the final signal is controlled by the gain variable. When the gain is less than or equal to 0, the wave component does not generate a signal.

When the waveform type is square, there is an additional control for the Mark/Space ratio. This ratio is the proportion of time, the square wave amplitude is positive to time, the square wave amplitude is negative in a given cycle. If the waveform type is not square, the Mark/Space ratio is ignored.

Audio Inputs		
FreqMod-Signal	Type	Default Value
	audio	n/a
Description: Frequency modulation signal, controls the actual generated frequency via the following formula: $\text{ActualFreq} = \text{Freq} \times (1 + (\text{ModDepth} \times \text{ModSignal}))$ Usually it falls in the range of 0 to 1.0, when used in conjunction with a unity gain modulation signal. Note: To avoid unpredictable behavior care should be taken to ensure that the product of modulation depth and modulation signal does not span a range greater than -1.0 to +1.0.		

Control Inputs		
Frequency	Type	Default Value
	float32	1.0
Description: Frequency (in Hertz) of wave generated by the waveform synthesizer. The frequency is the number of oscillations made per second for the given waveform. If not external variable is connected to Frequency, the value of the scaler is used. Modifier: Multiply Modifier_default: 0.0 Range: 0.0 -1/2*Sample Rate Units: Hz		

Control Inputs		
FreqMod-Depth	Type	Default Value
	float32	1.0
	Description: Frequency modulation depth value, controls the effect of the frequency modulation signal: $\text{ActualFreq} = \text{Freq} \times (1 + (\text{ModDepth} \times \text{ModSignal}))$ Usually it falls in the range of 0 to 1.0, when used in conjunction with a unity gain modulation signal. If no external variable is connected to FreqModDepth, the value of the scaler is used as the frequency modulation depth. Note: To avoid unpredictable behavior care should be taken to ensure that the product of modulation depth and modulation signal does not span a range greater than -1.0 to +1.0. Modifier: Multiply Modifier_default: 0.0 Range: 0.0 - 1.0	
Gain	Type	Default Value
	float32	1.0
	Description: Amplitude gain of the waveform. If no external variable is connected to Gain, the value of the scaler is used. Modifier: Multiply Modifier_default: 0.0 Range: 0.0 - Inf	

Control Inputs		
MarkSpaceInput	Type	Default Value
	float32	1.0
	Description: The mark/space input controls the duration the square wave is on, the plus side relative to the period of the wave. (e.g. a value of 0.5 means in a given period, the square wave output will be +Gain for half of the time and -Gain for the other half). If no external variable is connected to MarkSpace, the value of the scaler is used. Modifier: Multiply Modifier_default: 0.5 Range: 0.0 - 1.0	

Audio Output		
OutSignal	Type	Default Value
	audio	n/a
	Description: OutSignal is the output signal from the Wave component, which maybe connected to another component or directed to an output gateway.	

Internal Parameters		
Mark-SpaceUnits	Type	Default Value
	width_units	duty_cycle
	Description: This parameter is for the pulse and square inverse and defines the units for mark space input. The duty cycle range is 0-1. The value can also be set in seconds.	

Internal Parameters		
Wavetype	Type	Default Value
	audio/waveshape	sine
	Description: The type of the generated waveform, whether it is triangle, square sawtooth or sine. Range: Off, Sawtooth, Triangle, Sine, Square	

3.0. AudioIO

The AudioIO components are used to retrieve audio from the Highway 3D service. Typically, these components are linked directly to one of the hardware components (ACU, etc.).

The AudioIO components include:

- Headphone3DOut
- Highway Out
- SpeakerOut

These components are described in the Highway 3D Service section.

4.0. Comm Panel

The following section details the CommPanel components and the objects within them.

The CommPanel components include:

- CommPanel4
- CommPanel8
- CommPanel16
- CommPanel32
- CommPanel8Stereo
- StereoCommPanel

4.1. Comm Panel 4, 8, 16, 32

Summary: Simulation of a generic communication control panel used to link a single operator to up to 32 assets. A facility is provided for separate control of input, output and sidetone routing to and from assets. Assets are linked to the comm panel component using the intercom service.

Description: Although external assets may be connected, only a single channel needs to be described since the remainder are simply copies of the first. The basic concept of the comm panel is the connection of an operator input (usually microphone, often via a vox object) through a PTT gate and gain stage with an optional control input and scaling factor via a control selector switch (in_control) to an intercom channel (bi-directional) provided by the ACE intercom service. The intercom channel may be connected to various other object types to provide connectivity to other audio objects, or may simply be used as a basic intercom to provide standard intercom voice communications.

The input is keyed active if the PTT is set (either by default value or external control) and the input audio is tagged as Active_Tx.

The return signal from the service is passed to the operator via an output control selector via an optional receive gain control and scaling factor.

The comm panel object determines how sidetone is computed for each asset depending on the setting of the 'sidetone local' mask. If the bit corresponding to a particular channel is set then gain is computed locally within the comm panel, otherwise sidetone is a function of the externally connected asset. Sidetone Local is usually used for intercom connections between operators, while sidetone remote is used for connections to radio objects. Remote sidetone will cause operators sharing an asset to all hear each others sidetone, while sidetone local will cause other operators to hear only transmitted sidetone.

The LocalGain Bypass mask parameter determines whether the receive gain control and scaling factors affect sidetone gain or not. The normal operation is that the sidetone gain IS affected by the corresponding channel receive gain values, as well as the main Outgain value. If LocalGain Bypass bit is true for a particular signal asset, its sidetone gain will be the product of the Sidetone Gain, the OutGain, and the Rx Gain for the particular asset.

Likewise normal operation for the Sidetone Control byte is for it to be AND'ed with the OutControl byte. Unless of course the LocalGain Bypass is set true, in which case for that Asset the mask is controlled only by the Sidetone Control byte.

The comm panel may be connected to Tx/Rx intercom service channels (typically voice signals to/from radios and intercom between crew, or Rx only signals (such as Nav receivers, tone sources and similar).

The duplex behavior of any asset may be set based on the setting of the Full Duplex input and/or modifier. Default operation is full duplex. Asset connection duplex behavior will always force local behavior, so connections to a normal radio will force half-duplex operation.

The OutGain control and scaling factor determines the overall receive signal gain.

The SidetoneGain control and scaling factor determines the overall sidetone signal gain.

Power connection state determines whether the object is operational or not. All operation disabled when power is off (FALSE).

Audio Inputs		
InSignal	Type	Default Value
	audio	n/a
	Description: Input audio signal that is going to be transmitted by the Comm Panel.	

Control Inputs		
InControl	Type	Default Value
	uint	255
	Description: Selects the intercom busses to transmit the In Signal (bitmask). For example, a value of 1 will transmit on Sig1, a value of 2 on Sig2, 4 on Sig3, 8 on Sig4, 16 on Sig5, and 255 on all busses. Modifier: AND Modifier_default: 255 Range: 0 -255	

Control Inputs		
InGain	Type	Default Value
	float	1.0
	Description: Scales the In audio signal. Modifier: Multiply Modifier_default: 1.0 Range: 0.0-1.0	
OutControl	Type	Default Value
	uint	255
	Description: Selects the busses to receive audio from (bitmask). For example, a value of 1 will transmit on Sig1, a value of 2 on Sig2, 4 on Sig3, 8 on Sig4, 16 on Sig5, and 255 on all busses. Modifier: AND Modifier_default: 255 Range: 0 -255	
Control Inputs		
OutGain	Type	Default Value
	float	1.0
	Description: Volume control that scales the received Out signal obtained after mixing the received signals from each intercom bus. Modifier: Multiply Modifier_default: 1.0 Range: 0.0 -1.0	

Control Inputs		
Power	Type	Default Value
	bool	
	Description: Controls the power of the comm panel. Modifier: XOR Modifier_default: TRUE	
PTT	Type	Default Value
	bool	FALSE
	Description: Controls comm panel audio transmission. Modifier: XOR Modifier_default: True	
SideControl	Type	Default Value
	uint	255
	Description: Selects the busses from which to receive the sidetone signal. For example, a value of 1 will transmit on Sig1, a value of 2 on Sig2, 4 on Sig3, 8 on Sig4, 16 on Sig5, and 255 on all busses. Modifier: AND Modifier_default: 255 Range: 0 -255	
Sidetone Gain	Type	Default Value
	float32	0.6
	Description: Scales the Side (sidetone) signal obtained after mixing the received sidetone from each IC bus.	

Control Inputs		
Sig1-SigN	Type	Default Value
	id	unassigned
	Description: Selects the intercom bus handle.	
Sig1_RxGain-SigN-RxGain	Type	Default Value
	float	1.0
	Description: Volume control that scales the signal received from each intercom bus prior to mixing.	

Audio Outputs		
OutSignal	Type	Default Value
	audio	
	Description: Output audio signal generated by mixing all the received signals from the actively selected IC busses.	
SideSignal	Type	Default Value
	audio	
	Description: Sidetone audio signal generated by mixing all the received sidetones from the actively selected IC busses.	

Internal Parameters		
SideGain-Control	Type	Default Value
	uint	255
	Description: Selects which sidetones are affected by the comm panel's received signal gains. When bit is high, sidetone volume for the intercom bus is multiplied by the appropriate SigNRxGain. Also when bit is high the SideControl value for the intercom bus becomes the logical AND of SideControl and OutControl.	
Sidetone Local	Type	Default Value
	uint	0
	Description: Selects which sidetones are generated locally vs. remotely. If more than one comm panel is sharing the same intercom bus, this control determines if the sidetone is set to the other panels.	

4.2. CommPanel8Stereo

Summary: Simulation of a stereo communication control panel used to link a single operator to up to 8 assets. A facility is provided for separate control of input, output and sidetone routing to and from assets. Assets are linked to the comm panel component using the intercom service. This component has separate controls for left and right output and sidetone signals.

Description: This is a generic CommPanel component with stereo function. See the CommPanel component description for more information.

Audio Inputs		
InSignal	Type	Default Value
	audio	n/a
	Description: Input audio signal that is going to be transmitted by the Comm Panel.	

Control Inputs		
Power	Type	Default Value
	boolean	True
	Description: Controls the power of the comm panel.	
PTT	Type	Default Value
	boolean	True
	Description: Controls comm panel audio transmission.	

Control Inputs		
InGain	Type	Default Value
	float32	1.0
	Description: Scales the In audio signal.	
OutGain	Type	Default Value
	float32	1.0
	Description: Volume control that scales the received Out signal obtained after mixing the received signals from each intercom bus.	
Sidet-oneGain	Type	Default Value
	float32	0.600
	Description: Scales the Side (sidetone) signal obtained after mixing the received sidetone from each IC bus.	
InControl	Type	Default Value
	uint8	255
	Description: Selects the intercom busses to transmit the In Signal (bitmask). For example, a value of 1 will transmit on Sig1, a value of 2 on Sig2, 4 on Sig3, 8 on Sig4, 16 on Sig5, and 255 on all busses.	
OutControlL	Type	Default Value
	uint8	255
	Description: Selects the busses from which to receive the Rx signals routed to the OutSignalL. For example, a value of 1 will receive on Sig1, a value of 2 on Sig2, 4 on Sig3, 8 on Sig4, 16 on Sig5, and 255 on all busses.	

Control Inputs		
OutControlR	Type	Default Value
	uint8	255
	Description: Selects the busses from which to receive the Rx signals routed to the OutSignalR. For example, a value of 1 will receive on Sig1, a value of 2 on Sig2, 4 on Sig3, 8 on Sig4, 16 on Sig5, and 255 on all busses.	
SideControlL	Type	Default Value
	uint8	255
	Description: Selects the busses from which to receive the sidetone signal routed to SideSignalL. For example, a value of 1 will receive on Sig1, a value of 2 on Sig2, 4 on Sig3, 8 on Sig4, 16 on Sig5, and 255 on all busses.	
SideControlR	Type	Default Value
	uint8	255
	Description: Selects the busses from which to receive the sidetone signal routed to SideSignalR. For example, a value of 1 will receive on Sig1, a value of 2 on Sig2, 4 on Sig3, 8 on Sig4, 16 on Sig5, and 255 on all busses.	
SidetoneLocal	Type	Default Value
	uint8	0
	Description: Selects which sidetone signals are generated locally vs. remotely. If more than one comm panel is sharing the same intercom bus, this control determines if the sidetone is shared with the other panels.	

Control Inputs		
SideGainControl	Type	Default Value
	uint8	255
	Description: Selects which sidetone signals are affected by the comm panel's received signal gains. When bit is high, sidetone volume for the intercom bus is multiplied by the appropriate SigN_RxGainL or SigN_RxGainR, and by outgain bytes which is masked by OutControl bytes.	
Sig1_RxGainL-Sig8_RxGainL	Type	Default Value
	float32	1.0
	Description: Volume control that scales the signal received from each intercom bus prior to mixing into left output signal.	
Sig1_RxGainR-Sig8_RxGainR	Type	Default Value
	Description: Volume control that scales the signal received from each intercom bus prior to mixing into right output signal.	
Sig1-Sig8	Type	Default Value
	id	unassigned
	Description: Selects the intercom bus handle.	

Audio Outputs		
OutSig-nalL	Type	Default Value
	audio	n/a
	Description: Left output audio signal generated by mixing all the received signals from the actively selected IC busses.	
OutSig-nalR	Type	Default Value
	audio	n/a
	Description: Right output audio signal generated by mixing all the received signals from the actively selected IC busses.	
SideSig-nalL	Type	Default Value
	audio	n/a
	Description: Left sidetone audio signal generated by mixing all the received sidetone signals from the actively selected IC busses.	
SideSig-nalR	Type	Default Value
	audio	n/a
	Description: Right sidetone audio signal generated by mixing all the received sidetone signals from the actively selected IC busses.	

4.3. Stereo Comm Panel

Created for a specific program, contact ASTi for details.

Audio Inputs		
In	Type	Default Value
	audio	n/a
	Description: Input audio signal that is going to be transmitted by the Comm Panel.	
Side	Type	Default Value
	audio	n/a
	Description: Sidetone audio signal generated by mixing all the received sidetones from the actively selected IC busses.	

Control Inputs		
InControl	Type	Default Value
	float	1.0
	Description: Selects the eight bank A intercom busses to transmit the InSignal.	
InGain	Type	Default Value
	float	1.0
	Description: Scales the In audio signal. Modifier: Multiply Modifier_default: 1.0 Range: 0.0-1.0	

Control Inputs		
OutControl	Type	Default Value
	uint	255
	Description: Selects the busses to receive audio from (bitmask). Modifier: AND Modifier_default: 255 Range: 0 -255	
OutGain	Type	Default Value
	float	1.0
	Description: Scales the received Out signal obtained after mixing the received signals from each IC bus. Modifier: Multiply Modifier_default: 1.0 Range: 0.0 -1.0	
Power	Type	Default Value
	bool	FALSE
	Description: Controls the power bus of the comm panel. Modifier: XOR Modifier_default: TRUE	
PTT	Type	Default Value
	bool	FALSE
	Description: Controls comm panel audio transmission. Modifier: XOR Modifier_default: True	

Control Inputs		
SideControl	Type	Default Value
	uint	255
	Description: Selects the busses from which to receive the sidetone signal. Modifier: AND Modifier_default: 255 Range: 0 -255	
SidetoneGain	Type	Default Value
	float	1.0
	Description: Scales the Side (sidetone) signal obtained after mixing the received sidetone from each IC bus. Modifier: Multiply Modifier_default: 0.6 Range: 0.0 -1.0	
Sig_RxGain	Type	Default Value
	float	1.0
	Description: Scales the signal received from each IC bus prior to mixing.	

Audio Outputs		
Out	Type	Default Value
	audio	n/a
	Description: Output audio signal generated by mixing all the received signals from the actively selected IC busses.	

Internal Parameters		
SideGainControl	Type	Default Value
	uint	0
	Description: Selects which sidetones are affected by the comm panel's received signal gains.	
SidetoneLocal	Type	Default Value
	uint	255
	Description: Selects which sidetones are affected by the comm panel's receive signal gains.	

5.0. Control Components

Control components provide the logic for driving the functions of the objects in the model.

This section provides the control components which include the following:

- BitToByte
- ByteToBit
- ByteMerger
- ByteSplitter
- Counter
- Delay
- Ident
- Incrementer
- IntCompare
- IntTable
- Latch
- LogicTable
- MathFunction
- NumToString
- PassThrough

5.1. BitToByte

Summary: The Bit to Byte component provides a mechanism for combining up to eight boolean controls into a single byte wide value.

Description: The first input (Input0) becomes the least significant bit while the eight input (Input7) becomes the most significant bit. Each boolean control may be inverted at the component input. A final control gate is applied to the derived byte at the component output. If this gate is false, the component outputs 0. If this gate is true, the component outputs the byte value.

Control Inputs		
Gain	Type	Default Value
	float32	1.0
	Description: Overall gain applied to the output result. If the gain connection is blank then the gain scaler is used as the gain value. Modifier: Multiply Modifier_default: 1.0	
Input (0-7)	Type	Default Value
	boolean	FALSE
	Description: Controls the boolean values to be assembled into an 8 bit byte. Modifier: XOR Modifier_default: False	

Control Outputs		
Output	Type	Default Value
	float32	0.0
	Description: Output result of bit to byte function, integer value, a rounding of the floating point result.	

5.2. ByteToBit

Summary: The ByteToBit component provides a mechanism for splitting input byte into its eight bit values.

Description: The ByteToBit component converts a single input byte into its eight individual bit values. The ByteToBit component also has a Control Input which acts as an on/off switch turning the decoding On when True and Off when False.

Control Input		
Byte	Type	Default Value
	uint8	0
	Description: A single byte input that can either be hard coded or dynamically set by a host control or another linked component. Valid range is from 0-255. If the Control input is set to True, then the input byte value will be decoded and the Output Bit will be appropriately switched to True or False.	
Control	Type	Default Value
	boolean	True
	Description: The Control input acts as an on/off switch for the ByteToBit component. A value of True enables the decoding of a Byte into its bits while a value of False disables the decoding and the output bits will default to their initialized state. This control input is useful in conjunction with the internal Gate0 through Gate7 primitives. The Gate primitives can be used to change the False default state of Output Bits to default to True. In this situation, one can create any combination of initial bit states. Turning Control input to false effectively returns this component to its default state and consequently the component inputs linked to the Output Bits can be changed back to their defaults.	

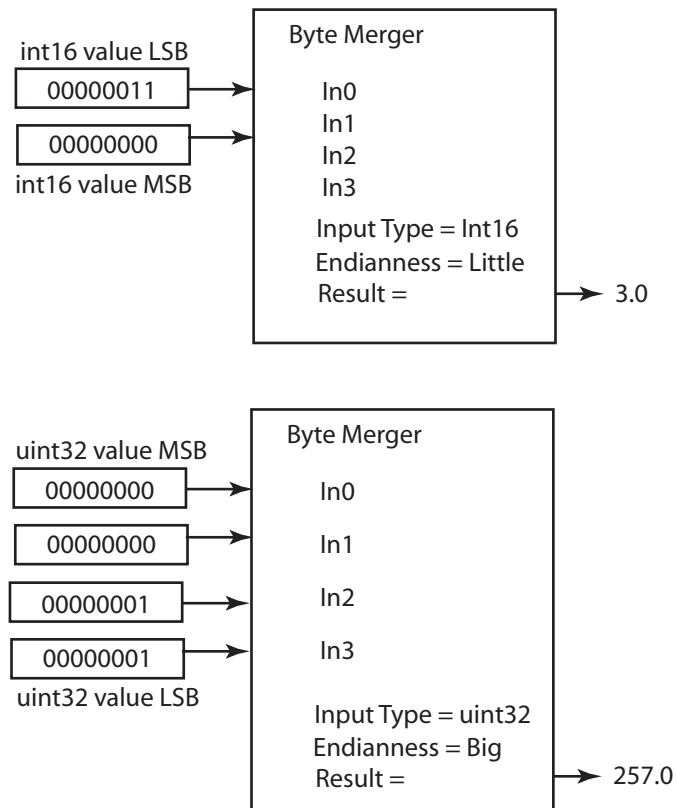
Control Input		
Gate0 through Gate7	Type	Default Value
	boolean	True
	Description: The Gate0 through Gate7 control inputs act as either buffers or inverters to each of their respective Bit Outputs depending on the state of the value variable. If the value variable in the Gate input is set to True, then the corresponding Output Bit is reverted. If the value variable is set to False, then the corresponding Output Bit will match its state based upon the Input Byte value.	

Control Output		
Bit0 through Bit7	Type	Default Value
	boolean	n/a
	Description: These are the 8 individual bit outputs that are the result of the input byte word and the Gate0 through Gate7 primitives. Bit0 is the least significant bit and Bit7 is the most significant bit.	

5.3. ByteMerger

Summary: The ByteMerger accepts 4 uint8 inputs, interprets them as a multibyte value (int16, uint16, int32, uint32, or float32) and outputs the result as a float64.

Description: The ByteMerger reconstructs an integer or float from a series of bytes. The input type is used to specify how the input byte should be integrated. For example, if reconstructing a two byte signed integer, link the least significant byte to In0 and the most significant byte to In1.



Control Inputs		
In0	Type	Default Value
	UInt8	0
	Description: The first byte of the multibyte value.	
In1	Type	Default Value
	UInt8	0
	Description: The second byte of the multibyte value.	
In2	Type	Default Value
	UInt8	0
	Description: The third byte of the multibyte value.	
In3	Type	Default Value
	UInt8	0
	Description: The fourth byte of the multibyte value.	

Control Outputs		
Endianness	Type	Default Value
	Endianness	Little
	Description: The byte order of the input bytes. If little endian then the least significant byte should be linked to In0. If big endian then the most significant byte should be linked to In0.	
Input Type	Type	Default Value
	types	UInt32
	Description: Specifies how the input bytes should be interpreted.	

Control Outputs		
Result	Type	Default Value
	float64	0.0
	Description: The result of merging the input bytes based on the endianness and input type.	

5.4. Byte Splitter

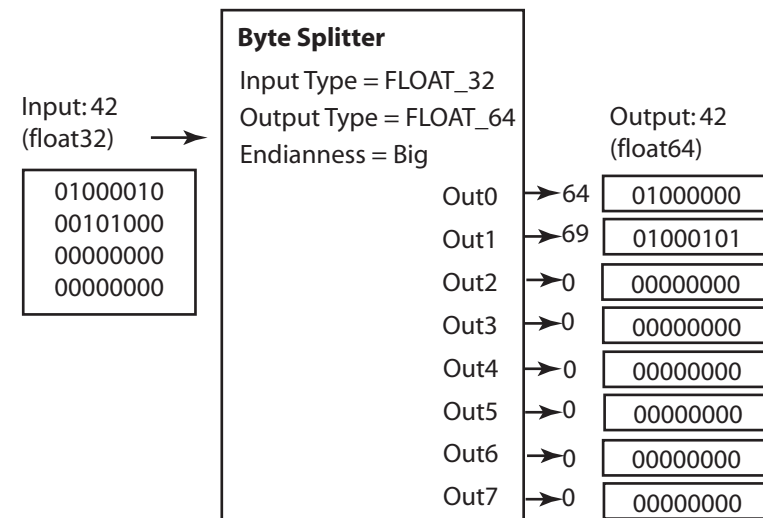
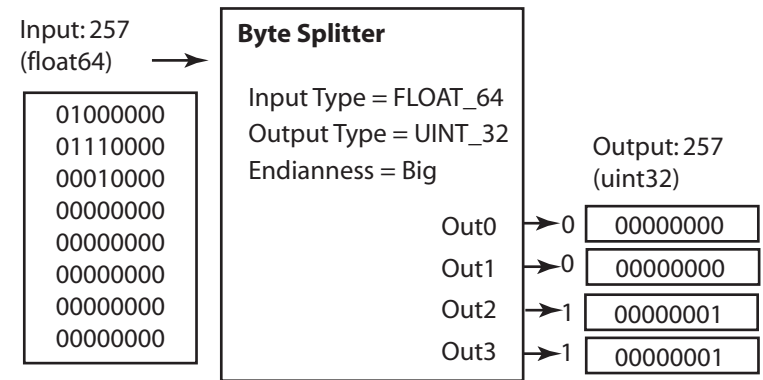
Summary: The Byte Splitter decomposes an input value into byte-sized chunks, allowing for datatype conversion at either end.

Description: An integral or float type input is accepted into the Byte Splitter, the user can select the specific input type. If an integral type is selected the component will only use the value from the InInt port. If a float type is selected the component will only use the value from the InFloat port. The unused value in either case will be set to 0.

InInt and InFloat are constrained to being represented as uint64 and float64 respectively. Depending on the actual datatype being input into the component (e.g. int64 linked to the uint64) the 'result' column value might be misrepresented. Although this might cause some confusion the user should rest assured that internally the component is aware of the actual representation of the input data (as specified by the InputType selector).

Examples*

* using IEEE floating point standard



Control Inputs		
InInt	Type	Default Value
	any integral type	0
	Description: Input port for integral-typed inputs.	
InFloat	Type	Default Value
	any float type	0
	Description: Input port for float-typed inputs.	

Control Outputs		
Out0-7	Type	Default Value
	UInt8	0
	Description: The output of the data in byte sized chunks. The size of the datatype determines how many output ports will actually be used.	

Internal Parameters		
Endian-ness	Type	Default Value
	Endianness	Little
	Description: Sets the endianness of the output data. Toggling this essentially reorders the data in the Out array.	
Input Type	Type	Default Value
	types2	INT_16
	Description: Selects which input port (InInt or InFloat) to use and lets the component know how to interpret the bit-level representation of the data coming in on the input port. This allows the user to input <i>any</i> integral or float type into the input ports.	
Output Type	Type	Default Value
	types2	INT_16
	Description: Allows the user to convert the input value to a different datatype before outputting the data in byte-sized chunks.	

5.5. Counter

Summary: The Counter provides a time-based, general purpose event or continuous ramping function. In single shot mode, the Counter can provide an externally triggered function lookup suitable for amplitude or frequency control of dynamically generated audio sources for sound effects such as explosions, or touchdown thumps that play once for a fixed time duration.

When set to continuous mode, the Counter can provide a table driven modulation of waveforms, where the modulation rate is slower than the overall model execution rate (i.e. 0 to 100 Hertz).

The Counter also provides a generic timer function for control logic within the model.

Description: The operation of the Counter component is as follows:

1. When triggered, the counter counts from Start Val to End Val over the period of time specified by Duration.
2. The current count value is passed as input X into an $f(x)$ function.
3. The output of the Counter component is the function result.

As described above, the Start Value and End Value fields define the counting sequence. If these parameters are set to the same value, the counter should output 0.0. If the End Value is less than the Start Value, the counter will count down. If the End Value is greater than the Start Value, the counter will count up. When the counter is not counting, its value is 0.0.

Note: The output of the Counter component may not necessarily be 0.0 when the counter is 0.0. The counter component output depends on the function that has been specified by the user.

If the Duration input is less than zero, the Counter should use a value of zero for the counter duration.

When the Continuous Mode flag is set, the counter continuously cycles through the count sequence. When it reaches the End Value, it waits for the period of time specified by Delay then begins the count sequence again. If this flag is not set, the counter resets to 0.0 when it reaches the end of the sequence.

When the Count All flag is set, the counter proceeds to the end of the count sequence even if the Trigger has been removed. At the end of the sequence, the counter resets to 0.0. If this flag is not set, the counter resets to 0.0 as soon as the Trigger is removed.

Control Inputs		
Delay	Type	Default Value
	float32	1.0
	Description: The length of time (in seconds) between counting cycles when the Counter is in Continuous Mode. Modifier: Multiply Modifier_default: 0.0	
Duration	Type	Default Value
	Float32	1.0
	Description: The length of time (in seconds) for the counter to go from Start_Value to End_Value. Modifier: Multiply	
Pause	Type	Default Value
	boolean	FALSE
	Description: The Counter will pause at its current value when Pause is True. Modifier: XOR	
Trigger	Type	Default Value
	boolean	FALSE
	Description: A Trigger state of TRUE initiates counting. The Continuous Mode flag determines how the counter behaves when the Trigger is set to False. Modifier: XOR Modifier_default:	

Control Outputs		
Result	Type	Default Value
	float32	0.0
	Description: This field reflects the final counter result.	

Internal Parameters		
Continuous	Type	Default Value
	boolean	FALSE
	Description: The Continuous Mode flag determines the Counter behavior when the Trigger is held TRUE. If this flag is TRUE, the Counter continually steps through the counting sequence. When it reaches the End Value, it returns to the Start Value and begins the counting sequence again. If this flag is FALSE, the Counter does only one pass through the counting sequence. It stops counting when it reaches the End Value.	
CountAll	Type	Default Value
	boolean	False
	Description: The Count All flag determines the Counter behavior when the Trigger transitions from TRUE to FALSE. If this FLAG is TRUE, the Counter continues counting until it reaches the End Value when the Trigger goes FALSE. If this FLAG is FALSE, the Counter immediately stops counting when the Trigger goes FALSE.	

Internal Parameters		
Function	Type	Default Value
	function handle	$f(x) = x$
	Description: The Function Service handle determines the final output based on the current counter value. The function may be of any of the types supported by the Function Service. The most commonly used counter functions are the Table and the Comparator. Modifier: Multiply Modifier_default: 1.0	
Function-Gain	Type	Default Value
	boolean	False
	Description: The FunctionGain is the overall gain applied to the Function output result.	
RangeEnd	Type	Default Value
	float32	1.0
	Description: This field specifies the ending value of the counter sequence. When triggered, the counter will count from Start_Val to End_Val.	
Range-Start	Type	Default Value
	float32	1.0
	Description: This field specifies the starting value of the counter sequence. When triggered, the counter will count from Start_Val to End_Val.	

Internal Parameters		
UpDown	Type	Default Value
	boolean	False
	Description: The Up/Down flag controls if the Counter should count from Start Val to End Val then back to Start Val within the period specified by Duration. This feature is useful for inserting windup and winddown effects to dynamically generated aural cue sounds.	

5.6. Delay

Summary: Delays the input control value.

Description: The Delay component delays a value by the specified number of K-Frames. Each K-Frame is 9.333 msec. The maximum settable delay is 107 K-Frames or approximately 1 second. This component can be used to delay control data (i.e. host data) within the model and may be useful for creating certain state logic.

Control Inputs		
DelayK-Frames	Type	Default Value
	int32	0
	Description: The number of K-Frames to delay the value. Each K-Frame is 9.333 msec. Maximum setting is 107 K-Frames or 1 second.	
Enable	Type	Default Value
	Boolean	False
	Description: When True the value is delayed. When False the value is not delayed.	
ValueIn	Type	Default Value
	Float32	0.0
	Description: An input value from another component in the model. This value is delayed and sent to ValueOut.	

Control Output		
ValueOut	Type	Default Value
	float32	0.0
	Description: The delayed value, which is based on ValueIn and delay K-Frames.	

Internal Parameters		
DelayMsec	Type	Default Value
	Float32	0
	Description: The K-Frame delay expressed in milliseconds.	

5.7. Ident

Summary: The Ident component, similar to the Radio > MorseKeyer component provides an interface between the HostIn packet and the model for any 4 character ASCII string identifier sequence.

Description: The Ident component decodes the incoming zero terminated ASCII string into the correct sequence of on/off pulses required for ident code communication. In addition to the usually letters and numbers defined in the Morse code, it also includes the characters (* and -) to represent individual dot and dash combinations. Offset to input variable from beginning of Ethernet packet, in bytes.

Control Inputs		
Carrier-Mode	Type	Default Value
	boolean	False
	Description: Provides control of carrier wave state gaps in identifier. When True the identifier has spaces appended to the front and back of the string, and the carrier wave is on when not keying Morse string.	
Ident	Type	Default Value
	ident	n/a
	Description: Controls the Ident from the host in.	
Inverted	Type	Default Value
	boolean	False
	Description: Provides local logic inversion of keying.	
Repeat-Rate	Type	Default Value
	uint16	1
	Description: Repeat period in seconds for retransmission of Morse code string. Modifier: 5	

Control Inputs		
WordRate	Type	Default Value
	uint8	1
	Description: Determines the rate at which the word is played. Units are in dots per second. The faster the rate the higher the number. Modifier: 8	

Control Outputs		
Result	Type	Default Value
	boolean	False
	Description: Output of morse code toggle tone.	

5.8. Incrementer

Summary: Tracks the number of times an event has occurred.

Description: The Incrementer component adds one to an output count when the trigger transitions, which can be used to keep track of basic state information. The output reverts to zero on model load or when the Reset flag is triggered. In addition to the reset flag, two other controls MaxValue and TimeToReset can be used to ‘auto-reset’ the output.

Control Inputs		
Reset	Type	Default Value
	boolean	False
	Description: Resets the output count to zero when set to true.	
Trigger	Type	Default Value
	int32	False
	Description: Increments the result by one on transition to a higher or lower number.	

Control Outputs		
Result	Type	Default Value
	uint16	0
	Description: The output count which increments by one when the trigger transitions.	

Internal Parameters		
MaxValue	Type	Default Value
	uint16	0
	Description: Acts as a ceiling for the result. When the result reaches MaxValue it is immediately set to zero. If MaxValue is left at zero, then the result will count up indefinitely.	
TimeToReset	Type	Default Value
	float32	0
	Description: Acts as a timer in seconds that begins every time the result is incremented. If the time expires before the next increment, the result is reset to zero. If TimeToReset is 0 then no timed reset will occur.	
Trigger Type	Type	Default Value
	TriggerType	RISING_EDGE
	Description: Determines when the count should increment. If set to RISING_EDGE, the result will increment when the trigger increases (0 to 1). If set to FALLING_EDGE, the result will increment when the trigger decreases (1 to 0). If set to ON_TRANSITION, the result will increment in either case.	

5.9. IntCompare

Summary: The IntCompare Component checks eight input values and compares them against a range of integers. Each integer is compared against a different range. The result of all eight comparisons is output as both a single byte bit mask and a boolean value.

Description: The IntCompare is a quick tool for comparing multiple values at the same time. For example, a single frequency can be compared against multiple frequency bands to determine if a radio's mode should be changed. The component is intended to replace the use of many IntTables or In-Range math functions.

Each input integer (InputA through H) is compared against its own range (RangeA through H). The Result is TRUE when all controlled inputs are within range. The Output also provides a bit mask so that the result of all 8 comparisons can be used within the model.

The ranges are all-inclusive.

Control Inputs		
InputA - H	Type	Default Value
	uint32	0
Description: This input integer is compared against its matching Range. For example, InputA is compared against RangeA, and InputB is compared against RangeB. The Range for each integer is inclusive. If the Input value is within range, it will set the corresponding bit in Output to 1. For example, if InputA is in range, the least significant bit of Output will be 1; if InputA is out of range it will be 0. InputB's comparison will be the second bit, and InputH will be the most significant bit. Modifier: Add Modifier_Default: 0		

Control Inputs		
Control-Byte	Type	Default Value
	uint8	255
Description: The ControlByte is a bit mask that determines which Input Integers are used for Result and Output. When equal to 255, all eight Inputs must be in range for Result to be TRUE. If equal to 1, only InputA and RangeA are used. Modifier: AND Modifier_Default: 255		

Control Outputs		
Output	Type	Default Value
	byte	255
Description: A bit mask that contains the result of all eight comparisons. Each bit of Output corresponds to one of the eight inputs. If the bit is 1, then the Input is within range. If the bit is zero, then the Input is out of range. For example, if InputA is within RangeA but the other seven Inputs are out of range, Output will be equal to 1.		
Result	Type	Default Value
	boolean	TRUE
Description: The master result of all comparisons. All controlled integers must be in range for Result to be TRUE. If the ControlByte is less than 255, only the integers selected need to be in range for Result to be TRUE.		

Internal Parameters		
Range Lower A-H	Type	Default Value
	uint32	0
	Description: The lower end of each range comparison. If the Input equals Range Lower, the comparison is “in range.” Each Input only looks at its corresponding Range Lower. InputA looks at RangeA, InputB looks at RangeB, and so on.	
Range Upper A-H	Type	Default Value
	uint32	0
	Description: The upper end of each range comparison. If the input equals Range Upper, the comparison is “in range.” Each Input only looks at its corresponding Range Upper. InputA looks at RangeA, InputB looks at RangeB, and so on.	

5.10. IntTable

Summary: The integer table provides a simple look up function for integer values. Such a function may be used to dynamically drive a sound file index to be played or receive/transmit selections in a communication panel.

Description: The integer table receives an integer input from an external connection and adds an offset to it to drive an index.

If the derived index is between 1 and 16 inclusive, the result is the corresponding entry in the lookup table. If the derived index is outside of this range, the result is the OutofRange value. The values in the lookup table are hard-coded as internal parameters.

This component then adds an offset to the result then multiplies that value by a gain multiplier. This ResultOffset input may be driven by an external connection. This input provides a means to effectively change all of the lookup table values by a fixed amount dynamically. This would be useful if the lookup value set should vary under certain logic conditions.

Finally, this component applies a gain multiplier to derive the output value.

Control Inputs		
Gain	Type	Default Value
	float32	1.0
	Description: Overall gain applied to the output result. If the gain connection is blank then the gain scaler is used as the gain value. Modifier: Multiply Modifier_default: 1.0	
Index	Type	Default Value
	int32	0
	Description: Index value used for table lookup. If no external variable is connected to Index, the value of the offset is used. Modifier: Add Modifier_default: 0	

Control Inputs		
ResultOffset	Type	Default Value
	int32	0
	Description: An offset value that is added to the value looked up in the table. If no external variable is connected to ResultOffset, the value of the offset modifier is used. Modifier: Add Modifier_default: 0	

Control Outputs		
Output	Type	Default Value
	float32	0.0
	Description: Final output value from the Integer Table component.	

Internal Parameters		
IndexNot1-16	Type	Default Value
	int32	0.0
	Description: Lookup table value for result index less than 1 and greater than 16.	
Lookup_Table	Type	Default Value
	int32	1-16
	Description: The lookup table value for result index 1-16	

5.11. Latch

Summary: The Latch Component takes an input value and holds the number for a specified time. The input can also be held indefinitely.

Description: The Latch Component is used for holding a control value inside the model for use by other components. This allows an input, which varies over time, to be stored temporarily and sent elsewhere in the model. The Latch is ideal for sampling a rapidly changing control value such as the Random Number math function, or for building state logic into the model in conjunction with the Incrementer component.

The component reads the Input value and sets it as the output value. The component will then hold that value for the duration of LatchTime. After LatchTime expires, a new Input value will be taken and again used as the Output. Alternatively, setting LatchTime to 0.0 will hold the Input value indefinitely once the Enable flag is set to TRUE.

Control Inputs		
Enable	Type	Default Value
	boolean	FALSE
	Description: Causes the component to hold and store the current input value. When Enable is FALSE, the output is always equal to the input. As soon as Enable turns TRUE, the Input value will be held as the Output value, even if Input changes state. If LatchTime is 0.0, the Output will not change until Enable turns FALSE. Modifier: XOR Modifier_default: FALSE	
Input	Type	Default Value
	float32	1.0
	Description: The control value to be latched and held. Enable must be TRUE for the input to be stored. Modifier: Multiply Modifier_default: 1.0	

Control Outputs		
Output	Type	Default Value
	float32	1.0
	Description: The result of the Latch. If Enable is FALSE, this value is equal to the Input. Otherwise, Output is equal to the value of Input at the time that Enable is set to TRUE. If LatchTime is non-zero and Enable is held to TRUE, the Output will re-latch after the duration of LatchTime. This will continue to happen as long as Enable is TRUE.	

Internal Parameters		
LatchTime	Type	Default Value
	float32	0.0
	Description: The time, in seconds, for the component to hold the input value before latching a new value. If Enable goes FALSE before LatchTime has completed, the Output will revert to the Input. A LatchTime of zero will cause the component to store the Input value as long as Enable is TRUE.	

5.12. LogicTable

Summary: The Logic table component provides a mechanism for combining up to four boolean controls into a single function. The four inputs are combined to form a 4-bit number which acts as an index into a 16-value array or lookup table. This array contains floating point values, so that a combination of control functions can be achieved in a simple fashion.

Description: The four boolean inputs into the component are combined into a single index value as follows:

$$\text{Index} = (\text{Input0} * 2^0) + (\text{Input1} * 2^1) + (\text{Input} * 2^2) + (\text{Input3} * 2^3)$$

The resulting index value is used as an index into the 16-value array. The array values are configured as parameters. The LoginTable component determines the array value at the given index. It then adds the array value to the chain value to derive the final output.

Control Inputs		
Chain	Type	Default Value
	float32	1.0
	Description: Value added to the gain scaled output from the lookup table. Modifier: Multiply Modifier_default: 0.0	
Gain	Type	Default Value
	float32	1.0
	Description: Scales the output from the lookup table. Modifier: Multiply Modifier_default: 1.0	

Control Inputs		
Input0	Type	Default Value
	boolean	False
	Description: Boolean input whose value is assigned to Bit0 in the derived index value. If this variable is True, a value of 1 is added to the derived index value. Modifier: XOR Modifier_default: FALSE	
Input1	Type	Default Value
	boolean	False
	Description: Boolean input whose value is assigned to Bit1 in the derived index value. If this variable is True, a value of 1 is added to the derived index value. Modifier: XOR Modifier_default: FALSE	
Input2	Type	Default Value
	boolean	False
	Description: Boolean input whose value is assigned to Bit2 in the derived index value. If this variable is True, a value of 1 is added to the derived index value. Modifier: XOR Modifier_default: FALSE	

Control Inputs		
Input3	Type	Default Value
	boolean	False
	Description: Boolean input whose value is assigned to Bit3 in the derived index value. If this variable is True, a value of 1 is added to the derived index value. Modifier: XOR Modifier_default: FALSE	

Control Output		
Output	Type	Default Value
	float32	0
	Description: Final output value from the Logic Table component.	

Internal Parameters		
Lookup_Table	Type	Default Value
	float32	0-15.0
	Description: Lookup table value for result index 0-15.	

5.13. MathFunction

Summary:

Background: The ACE Host Input component does not include built-in scaling features - it simply provides a means to extract control fields from incoming host packets. Scaling of the Host Input fields is done in two ways:

1. The Host Input (data source) is linked directly to an internal variable field of a component (data sink). The sink component includes a basic scaling feature including a single operand and modifier which modify the value of the Host Input.
2. The Host Input* is linked to a Math Function component, which is configured to modify the value of the Host Input.

*Actually up to three Host Inputs can be linked to a Math Function. Also, Math Function Inputs can be connected to the outputs of other Math functions, to provide grouped or nested functions. For simplicity, this description uses Host Input to mean "data source."

The Math Function calls a user-specified function (like: lookup table, add, subtract, multiply, divide, etc.) which acts on the input (or inputs). The output is the result of the specified function acting on the input variables.

The Math Function is then linked to a variable inside the data sink component.

Description: Schematically, the Math Function is connected to a series, between a Host Input data source and the sink component using the control data to modify one of its internal variables. The Math Function provides a variety of modifier features for the Host Input:

1. One, two or three data source inputs,
2. A link to the ACE Function Service, providing access to: one variable functions, or $f(x)$, two variable functions, or $f(x,y)$ and three variable functions, $f(x,y,z)$.

The Math Function has three inputs: X, Y and Z and serves as a one, two or three variable math functions. Here is how it works:

One variable function: link one input to a Host Input and a one variable functions, $f(x)$ is selected. The two unused inputs assume values of 1.

Two variable function: two inputs are linked to Host Inputs and a two variable function, $f(x,y)$ is selected. The unused input assumes a value of 1.

Three variable function: all three inputs are linked to Host Inputs and a three variable function, $f(x,y,z)$ is selected.

Each input has its own constant scaling factor and multiplier operand. Each of the input scaling factors are user selectable.

The function operating on the X, Y, and Z inputs is selected using the function handle. The function handle specifies an $f(x)$, $f(x,y)$ or $f(x,y,z)$ function from the Function Service. Function handles are defined in the tool associated with the Function Service. The Math Function includes a user interface for selecting a function handle (e.g. a pull-down list). Function handles include: Table (x), Scale/Limit (x), Log/Antilog (x), Lag Filter (x), Add/ Subtract/ Multiply/ Divide (x, y), Random Number (x), Comparator / MaxMin (x), Switch (x, y, z).

The evaluation of the function is modified by a user-selectable scaling factor and multiplier operand.

A Gain input is also available for use as a final-stage scaling factor - it modifies the scaled evaluation of the function. The Gain input has its own user-selectable constant scaling factor and multiplier operand.

The final result of Math Function is: the scaled Gain value multiplied by the scaled function evaluation. Result data types include:

- Float: Output result of math function, floating point value.
- Integer: Output result of math function, integer value, a rounding of the floating point result.
- Boolean: Output result of math function, as a boolean. The boolean (On/Off) is a digital comparison of the float value based on a 0.3 and 0.7 low and high threshold value. Below 0.3 is Off, above 0.7 is On, the 0.4 difference provides a hysteresis value.

Control Inputs		
Gain	Type	Default Value
	float32	1.0
	Description: Link to another control source to provide overall gain control of the math function evaluation. If no external variable is connected to Gain, the value of the scaler is used. Modifier: Multiply Modifier_default: 1.0 Range: min. - max. values of each type	
Input_X	Type	Default Value
	float32	1.0
	Description: Link to another control source (Host Input or Math Function) which provides the first variable to be used by the math function (first stack element). If no external variable is connected to Input_X, the value of the scaler is used. Modifier: Multiply Modifier_default: 1.0 Range: min. - max. values of each type	
Input_Y	Type	Default Value
	float32	1.0
	Description: Link to another control source (Host Input or Math Function) which provides the second variable to be used by the math function (second stack element). If no external variable is connected to Input_Y, the value of the scaler is used. Modifier: Multiply Modifier_default: 1.0 Range: min. - max. values of each type	

Control Inputs		
Input_Z	Type	Default Value
	float32	1.0
	Description: Link to another control source (Host Input or Math Function) which provides the third variable to be used by the math function (third stack element). If no external variable is connected to Input_Z, the value of the scaler is used. Modifier: Multiply Modifier_default: 1.0 Range: min. - max. values of each type	

Control Output		
Result	Type	Default Value
	float32	1.0
	Description: The result of the input values, acted on by the specified function, internal multiplier scaling factors.	

Internal Parameters		
Function	Type	Default Value
	function	f(x,y)=x
	Description: The function service handle selections include: Table (x), Scale (x,y), Random Number (x), Comparator /MaxMin (x), Switch (x, y, z). Range: Table, scale, limit, log, antilog, lag filter, add, subtract, multiply, divide, random, comparator, maxmin, switch.	

5.12. NumToString

Summary: The NumToString component takes a set of input numbers and converts them into a single string for use with the Radio Transceiver. This allows host control of common DIS parameters for radios by setting the DomainNameIn and ProtocolIDIn fields of the Radio Transceiver.

Description: NumToString takes up to four input numbers and concatenates them together to form a string. A prefix, suffix, and character separator are included to help fit the ProtocolID and DomainName syntax for radios. The most common use of this component is to set the Exercise ID of the radio by creating a string such as:

DIS:5

The string above can be used in the Domain Name In field of the Transceiver component and will set the DIS Exercise ID of the radio to 5. With this method Input0 is set to 5, the Prefix is set to "DIS:" and the InputCount is set to 1. Input0 can also be set dynamically from another component, allowing the Exercise ID of the radio to be changed as needed.

The NumToString is also used to produce strings such as:

DIS:80.1.3.17

This string can be used in the Protocol ID In Field of a Transceiver and will set the Site ID = 80, Application ID = 1, Entity ID = 3, and the Radio ID = 17. With this method, Input0 through 3 can be dynamically set from another component or Host In, allowing users to change any of the DIS IDs for a radio on an as needed basis. In order to create the string above, the prefix should be set to "DIS:" and the InputCount should be set to 4. Additionally, Input0 through 3 are set to digits used above and a period is used for the separator.

Control Inputs		
Input0	Type	Default Value
	int32	0
	Description: This input turns into part of the output string based on the conditional parameters. If used for Domain Name: Domain Name = DIS Exercise ID If used for Protocol ID: Protocol ID = DIS Entity ID or DIS Site ID	
Input1	Type	Default Value
	int32	0
	Description: This input turns into part of the output string based on the conditional parameters. Used for: Protocol ID = DIS Radio ID or DIS Application ID	
Input2	Type	Default Value
	int32	0
	Description: This input turns into part of the output string based on the conditional parameters. Used for: Protocol ID = DIS Entity ID	

Control Inputs		
Input3	Type	Default Value
	int32	0
	Description: This input turns into part of the output string based on the conditional parameters. User for: Protocol ID = DIS Radio ID	

Control Output		
Output	Type	Default Value
	string	0000
	Description: Outputs the string values, maximum number of string characters is 32.	

Internal Parameters		
Input-Count	Type	Default Value
	int32	0
	Description: Sets the number of values that are defined by the inputs, typically 4.	
KCycles	Type	Default Value
	int32	0
	Description: Currently not required.	

Internal Parameters		
Prefix	Type	Default Value
	string	n/a
	Description: Attaches to the front of the string. Typically used for “DIS” or colon “:”.	
Separator	Type	Default Value
	string	0
	Description: Attaches to the string to separate the values. Typically used for “.” or space “ ”.	
Suffix	Type	Default Value
	string	n/a
	Description: Attaches to the end of the string and may be any string value.	

5.13. PassThrough

Description: The VarPassThrough component is used to pass variables from a HostIn component to a HostOut component. It is not designed for any other purpose and should not be used otherwise. An example of a use case for this component is to transfer unmodified host input variables through the model and out to a separate host PC or touch screen.

6.0. Dynamics

The following section details the Dynamics components and the objects within them, which are used to control the volume or level of signals.

The Dynamics group includes the following components.

- The AGC component keeps a signal near a “target” level.
- The CompressorLimiter component prevents signals from getting too loud.
- The Expander component reduces the volume of quiet signals like background noise between speech.
- The Gate component only allows a signal through if it is loud enough.

Common to all four of these components are controls measured in dB and dB/sec. The dB controls such as target and threshold specify signal levels relative to 1.0 (as seen in the signal scope). The following table shows a few dB values and their linear scale equivalents.

dB	Linear
0	1.0
-6	0.5
-12	0.25
-18	0.125
---	---
-40	0.01
-60	0.001

Speech, for example, might be around -20dB in the system (depending on the microphone and input gains) while background noise might be around -60dB. Attack and release gain controls are measured in dB/sec, which is one way of specifying how quickly the gain can change. An attack of 6dB/sec, for example, would mean the gain could double every second. Generally, practical attack and release rates are much quicker, typically in the range of 50-1000 dB/sec.

6.1. AGC

Summary: AGC attempts to keep audio at a consistent, specified volume.

Description: The Automatic Gain Control (AGC) component controls the volume of the signal to keep the output close to a target volume. There are two stages in this component, the AGC and the Limiter. In the AGC stage, signals above the target are reduced, while signals below the target (and above the threshold) are increased. The Limiter stage then acts to prevent any high peaks that may have been induced by the AGC.

This component can be useful for a variety of volume leveling tasks, such as evening out microphone volumes before the input of the speech recognition system.

Audio Input		
InSignal	Type	Default Value
	audio	n/a
	Description: The signal that will be filtered. This input is linked into the component from somewhere else in the model.	

Control Inputs		
Attack	Type	Default Value
	float32	250.0
	Description: Determines how quickly the gain is turned up when the level goes below the target. Values are in dB/sec. Higher attacks produce a more aggressive rate of gain increase.	
Enable	Type	Default Value
	boolean	True
	Description: Controls whether any gain adjustment will occur.	

Control Inputs		
LimEnable	Type	Default Value
	boolean	True
	Description: Turns the limiter stage on or off in the component. If True, the peak output levels of the AGC will not go above the LimThreshold.	
LimThreshhold	Type	Default Value
	float32	-18.0
	Description: The maximum peak level in dB allowed in the output signal. -18 corresponds to 0.125 linear.	
OutGain	Type	Default Value
	float32	1.0
	Description: A gain applied to the signal after the AGC stage.	
Ratio	Type	Default Value
	float32	4.0
	Description: Controls how much the signal level is pushed towards the target. A ratio of 4.0 (4:1) means a signal of 4 dB from the target will leave the AGC only 1 dB away. Higher ratios have more “push”.	
Release	Type	Default Value
	float32	250
	Description: Determines how quickly the gain is turned down when the signal level goes above the target. Values are in dB/sec. Higher releases produce a more aggressive rate of gain reduction.	

Control Inputs		
Target	Type	Default Value
	float32	-24.0
	Description: The target or “goal” output signal level (RMS), measured in dB, relative to a level of 1.0. As with any dB control, more negative values are quieter, larger values are louder.	
Threshold	Type	Default Value
	float32	-70.0
	Description: The level above which AGC will occur. This should be adjusted to be just above the noise floor, so background noise is not boosted in volume. Values are in dB, relative to an RMS level of 1.0.	

Audio Output		
OutSignal	Type	Default Value
	audio	n/a
	Description: The processed signal that has gone through AGC and limiter stages.	

6.2. CompressorLimiter

Summary: The CompressorLimiter reduces the volume of loud sounds.

Description: This component has two stages that work together to keep volume (signal levels) under control. In the first stage, the compressor reduces the volume of the signal when its average (RMS) level exceeds the threshold. In the second stage, the limiter then prevents peak levels from exceeding the LimThreshold. This makes it a useful tool for preventing sound levels or clipping that could cause damage to equipment or your hearing.

Audio Input		
InSignal	Type	Default Value
	audio	n/a
	Description: The signal that will be filtered. This input is linked into the component from somewhere else in the model.	
SideChain	Type	Default Value
	audio	n/a
	Description: A second audio input which is used for level calculations if SideChainEnable is True. In this mode, the InSignal is compressed only if the sidechain exceeds the threshold. This is useful for controlling the relative volumes of two sounds.	

Control Input		
Attack	Type	Default Value
	float32	350.0
	Description: Determines how quickly the gain is turned down when the signal level goes above the threshold. Values are in dB/sec. Higher attacks produce a more aggressive rate of gain reduction.	

Control Inputs		
Comp OutGain	Type	Default Value
	float32	1.0
	Description: A gain applied to the signal after the compressor stage but before the limiter stage.	
Enable	Type	Default Value
	boolean	True
	Description: Controls whether the compressor stage will do any gain adjustment.	
LimAllow Clip	Type	Default Value
	boolean	False
	Description: If True, the signal peaks will be hard clipped at the LimThreshold level. This can be used to produce radio distortion-like effects.	
LimEnable	Type	Default Value
	boolean	True
	Description: Turns the limiter stage on or off in the component. If True, the peak output levels of the component will not go above the LimThreshold.	
LimRelease	Type	Default Value
	float32	100.0
	Description: The rate, in dB/sec, that determines how quickly the limiter responds to a drop in signal levels. Higher releases cause a faster increase in gain to ensure quiet signals are not reduced in volume.	

Control Inputs		
LimThresh	Type	Default Value
	float32	-12.0
	Description: The maximum peak level in dB allowed in the output signal. -12 corresponds to 0.25 linear, or 1/4 of a “full scale” signal.	
OutGain	Type	Default Value
	float32	1.0
	Description: A gain applied to the signal after both the compressor and limiter stages.	
Ratio	Type	Default Value
	float32	4.0
	Description: Controls the aggressiveness of the compressor. A high ratio is more aggressive and means more gain reduction will occur when the signal goes above the threshold. Specifically, a ratio of 4.0 (4:1) means a signal of 4 dB above the threshold will leave the component only 1 dB above the threshold. A ratio of 20 or higher is equivalent to (∞ :1), meaning the average level of the signal will never exceed the threshold.	
Release	Type	Default Value
	float32	50.0
	Description: Determines how quickly the gain is turned up when the level goes below the threshold. Values are in dB/sec. Higher releases produce a more aggressive rate of gain increase.	
Side-Chain Enable	Type	Default Value
	boolean	False
	Description: If True, the SideChain audio input will control the gain reduction on the InSignal, rather than the InSignal itself.	

Control Inputs		
Threshold	Type	Default Value
	float32	-24.0
	Description: The average (RMS) signal level above which gain reduction will take place. Values are in dB.	

Audio Output		
OutSignal	Type	Default Value
	audio	n/a
	Description: The processed signal that has gone through the compressor and limiter stages.	

6.3. Expander

Summary: The Expander component is used to reduce the volume of background noise between speech.

Description: The Expander expands (increases) the dynamic range of a signal by making quiet sounds quieter. If the average (RMS) level of the signal is below the threshold, the expander will smoothly reduce the gain on the signal. The key to using the Expander is to adjust the threshold to be just above the noise floor. This component is useful in hot mic situations to reduce background noise transmission.

Audio Input		
InSignal	Type	Default Value
	audio	n/a
	Description: The signal that will be filtered. This input is linked into the component from somewhere else in the model.	

Control Input		
Attack	Type	Default Value
	float32	250.0
	Description: Attack controls how quickly the gain changes in response to an increase in signal level. For the expander, this means it controls how quickly gain is restored to 1.0 when speech resumes.	
Enable	Type	Default Value
	boolean	True
	Description: Controls whether any gain adjustment will occur.	

Control Inputs		
OutGain	Type	Default Value
	float32	1.0
	Description: Gain applied after the expander stage.	
Ratio	Type	Default Value
	float32	2.0
	Description: Controls how aggressively sounds below the threshold are reduced in volume. A ratio of 2.0 (2:1) means a signal of 1 dB below will exit the component 2 dB below.	
Release	Type	Default Value
	float32	50.0
	Description: Release controls how quickly the gain changes in response to a decrease in signal level. High releases will cause a quick reduction in background noise when speech stops.	
Threshold	Type	Default Value
	float32	-55.0
	Description: The level in dB below which expansion (gain reduction) will take place. Adjust the threshold to be just above the loudest sound that should be reduced in volume. Often times this corresponds to just above the background noise level.	

Audio Output		
OutSignal	Type	Default Value
	audio	n/a
	Description: The processed signal which has gone through the expander stage.	

6.4. Gate

Summary: The Gate component only allows a signal through if it is loud enough to break the threshold.

Description: This component is similar to the Audio/Vox component in that it is used to only let speech through (to a radio, etc.) if a person is actively speaking. In this example, the threshold should be adjusted to be just below the quietest speech that is allowed through. The Gate component reacts to peak levels in the signal, unlike the Vox which looks at average levels. This makes the Gate more responsive, especially to the first sounds as speech starts.

Audio Input		
InSignal	Type	Default Value
	audio	n/a
	Description: The signal that will be filtered. This input is linked into the component from somewhere else in the model.	

Control Input		
Depth	Type	Default Value
	float32	40.0
	Description: The amount of gain reduction in dB that is applied when the signal dips below the threshold.	
Enable	Type	Default Value
	boolean	True
	Description: Controls whether any gain adjustment will occur.	

Control Inputs		
Hold	Type	Default Value
	float32	1000.0
	Description: A delay in milliseconds before reducing gain on a signal after it goes below the threshold. This allows the signal through the Gate for the hold time, even though it is below the threshold.	
OutGain	Type	Default Value
	float32	1.0
	Description: A gain applied after the Gate stage.	
PTT	Type	Default Value
	boolean	False
	Description: A control used to open the Gate (let the signal through), even if it is below the threshold.	
Threshold	Type	Default Value
	float32	-45.0
	Description: The level in dB below which a signal will be attenuated in volume or not allowed through the Gate.	
VoxMode	Type	Default Value
	boolean	False
	Description: If True, the Gate will act like the Vox component and cause the audio stream to go inactive if it is below the threshold. If False, signals below the threshold will be attenuated (according to the depth setting) but will remain active.	

Audio Output		
OutSignal	Type	Default Value
	audio	n/a
	Description: The processed signal that has gone through the gate stages.	

7.0. Environmental Cue

The following section details the environmental cue components and the objects within them.

The environmental cue components include:

- 5BandFilter
- Engine
- Engine Level D
- MultiFilter
- PropRotor
- VibrationCapture

7.1. 5BandFilter

Summary: Combines 5 MultiFilters into one component.

Description: This component essentially connects 5 MultiFilters in serial. This component therefore consumes more processing power and should be used for complex filtering schemes. The advantage of using this component over 5 MultiFilters is that all inter-filter routing is handled within the component.

See MultiFilter description in this document for more information.

Audio Inputs		
InSignal	Type	Default Value
	audio	n/a
	Description: The signal to be filtered. This input should be linked into the component from somewhere else in the model.	

Control Inputs		
Enable1-5	Type	Default Value
	boolean	False
	Description: This control determines if the signal is filtered. If this control results in False the signal is not filtered but the OutGain control is still applied. Modifier: XOR Modifier_default: False	

Control Inputs		
Frequency 1-5	Type	Default Value
	float32	1.0
	Description: This control specifies frequency in Hertz. For the BandPass, BandPassUnityGain, and PeakingEQ filters this control specifies the center frequency of the passband. For the Notch filter this control specifies the center frequency of the stopband. For the LowPass and HighPass filters this control specifies the corner (-3 dB) frequency. For the LowShelf and HighShelf filters this control specifies the midpoint frequency. The range of this control is [20, 24000). Modifier: Multiply Modifier_default: 500	
Gain_dB1 -5	Type	Default Value
	float32	0.0
	Description: This control specifies a gain value in dB. For Peak-ingEQ, HighShelf, and LowShelf filters this control specifies the gain applied to the passband. The range of this control is [-50,50]. Modifier: Plus Modifier_default: 0.0	
OutGain_ 1-5	Type	Default Value
	float32	0.0
	Description: This control specifies gain in a linear scale. For all filters this control specifies the gain applied to the post-filtered signal. This control takes effect even if Enable results in False and if FilterType is 'Off'. The range of this control is [0, 316.2277] and a setting of 1.0 corresponds to a gain of 0 dB. Modifier: Multiply Modifier_default: 1.0	

Control Inputs		
QFactor1- 5	Type	Default Value
	float32	1.0
	Description: The filter quality factor. This control specifies the bandwidth for BandPass, BandPassUnityGain, Notch, and Peak-ingEQ filters; the higher this value, the smaller the bandwidth. The StartBand and EndBand variables adjust according to the QFactor and are not adjustable in the display. For HighPass, LowPass, High-Shelf, and LowShelf filters this control specifies the roll-off / corner frequency gain; the higher this value the steeper the roll-off and the higher the gain at the corner frequency. The range of this control is (0, 16) and setting this control to a value of 0 or lower results in the control defaulting to 0.707107. Modifier: Multiply Modifier_default: 0.7071.	

Audio Outputs		
OutSignal	Type	Default Value
	audio	n/a
	Description: The signal after it is filtered.	

Internal Parameters		
EndBand1 -5	Type	Default Value
	float32	965.4772
	Description: Specifies the end of the bandwidth affected by the filter settings. Not adjustable. For BandPass, BandPassUnityGain, and Notch filters this specifies the right -3 dB corner frequency in hertz. For PeakingEQ filters this specifies the midpoint frequency in hertz.	
FilterType 1-5	Type	Default Value
	filter_type3	LowPass
	Description: FilterType determines the type of filter applied to the input signal. If no filtering is desired, FilterType should be set to Off.	
Order1-5	Type	Default Value
	filter_order	_12dB_Per_Octave
	Description: FilterOrder determines the order of the filter applied to the input signal. 2 nd , 4 th , and 6 th orders are available, corresponding to 12, 24, and 36 dB per octave roll-off in LowPass, HighPass, LowShelf, and HighShelf filters and 6, 12, and 18 dB per octave roll-off in BandPass, BandPassUnityGain, Notch and PeakingEQ filters (assuming QFactor is fixed at 0.7071). Generally, the higher this control the steeper the roll-offs will be.	
StartBand 1-5	Type	Default Value
	float32	258.9393
	Description: Specifies the start of the bandwidth affected by the filter settings. Not adjustable. For BandPass, BandPassUnityGain, and Notch filters this specifies the left -3 dB corner frequency in hertz. For PeakingEQ filters this specifies the midpoint frequency in hertz.	

7.2. Engine

Summary: The engine component can recreate the tones for a single jet engine.

Description: The Engine component provides a composite sound for an engine. The component includes two principal sources for noise and three independent whine tones.

The overall sound can be tuned based on manipulating the driving parameters for the noises, whines, and overall gain control.

Control Inputs		
NoiseFilter 1	Type	Default Value
	float32	1.0
	Description: Provides control connections for one of the two noise gains.	
NoiseFilter 1_GainInput	Type	Default Value
	float32	1.0
	Description: GainFunction: Connections to a selected table or function for controlling the noise roll-off frequency based upon the input control connections. GainFunctionScalar: Scaling factors for each noise gain control. GainResult: The final gain factor for one of the two filters used.	

Control Inputs		
NoiseFilter 1_FrequencyInput	Type	Default Value
	float32	1.0
	Description: FrequencyFunction: Provides frequency control connections for one of the two bandwidth limited noise sources. FrequencyFunctionScalar: Scaling factors for each noise frequency control. FrequencyResult: The roll-off frequency (in Hertz) of one of the two bandwidth limited noise sources used to produce the engine hiss/roar.	
NoiseFilter 2	Type	Default Value
	float32	1.0
	Description: Provides control connections for one of the two noise gains.	
NoiseFilter 2_GainInput	Type	Default Value
	float32	1.0
	Description: GainFunction: Connections to a selected table or function for controlling the noise roll-off frequency based upon the input control connections. GainFunctionScalar: Scaling factors for each noise gain control. GainResult: The final gain factor for one of the two filters used.	

Control Inputs		
NoiseFilter2_FrequencyInput	Type	Default Value
	float32	1.0
	Description: FrequencyFunction: Provide frequency control connections for one of the two bandwidth limited noise sources. FrequencyFunctionScalar: Scaling factors for each noise frequency control. FrequencyResult: The roll-off frequency (in Hertz) of one of the two bandwidth limited noise sources used to produce the engine hiss/roar.	
OutGain	Type	Default Value
	float32	1.0
	Description: The OutGain applies amplitude gain control to the output signal. If no external control is connected to OutGain, the scale factor is used as the OutGain values.	
WhineTone1	Type	Default Value
	Description: Provides control connection for one of the three whine tones.	

Control Inputs		
WhineTone1_GainInput	Type	Default Value
	float32	1.0
	Description: Provides control connections for one of the three whine gains. GainFunction: Connections to a selected table or function for controlling the whine gains based upon the input control connection. GainFunctionScalar: The scaling factors for the whine gain control GainResult: The final gain factor for one of the three triangle waves used to produce the engine whines.	
WhineTone1_Frequency	Type	Default Value
	float32	1.0
	Description: The frequency (in Hertz) of one of the three triangle waves used to produce the engine whines.	
WhineTone2	Type	Default Value
	Description: Provides control connection for one of the three whine tones.	
WhineTone2_GainInput	Type	Default Value
	float32	1.0
	Description: Provides control connections for one of the three whine gains. GainFunction: Connections to a selected table or function for controlling the whine gains based upon the input control connection. GainFunctionScalar: The scaling factors for the whine gain control GainResult: The final gain factor for one of the three triangle waves used to produce the engine whines.	

Control Inputs		
Whine Tone2_Fre quency	Type	Default Value
	float32	1.0
	Description: The frequency (in Hertz) of one of the three triangle waves used to produce the engine whines.	
Whine Tone3	Type	Default Value
	Description: Provides control connection for one of the three whine tones.	
WhineTon e3_GainIn put	Type	Default Value
	float32	1.0
	Description: Provides control connections for one of the three whine gains. GainFunction: Connections to a selected table or function for controlling the whine gains based upon the input control connection. GainFunctionScalar: The scaling factors for the whine gain control GainResult: The final gain factor for one of the three triangle waves used to produce the engine whines.	
Whine Tone3_Fre quency	Type	Default Value
	float32	1.0
	Description: The frequency (in Hertz) of one of the three triangle waves used to produce the engine whines.	

Audio Output		
Output Signal	Type	Default Value
	audio	True
	Description: The output signal from the Engine component, which may be connected to another component or directed to an output highway.	

7.3. Engine Level D

Summary: A high fidelity engine component that can recreate intake hisses, combustion roar, blade and tip buzz, and characteristic whine tones.

Description: The Engine Level D component provides a composite sound for engines at the highest fidelity level required by the FAA. The component includes five principal sources for noise including: the impact of the blade movement through air, the effects of air velocity, the noise effect based on aircraft elevation, the noise effect of engine fuel consumption, and the noise effect of engine ignition.

The overall sound can be tuned based on manipulating the RPM, airspeed, fuel flow, and overall gain control.

Control Inputs		
AirSpeed	Type	Default Value
	float32	1.0
	Description: Provides the control of the airspeed which drives the intake hiss sound.	
Altitude	Type	Default Value
	float32	1.0
	Description: Provides a gain effect based on altitude, input effects the overall output of the component.	
EngineLit	Type	Default Value
	float32	1.0
	Description: This controls when the igniters light the fuel in the engine.	
FuelFlow	Type	Default Value
	float32	1.0
	Description: The rate of fuel consumed by the engine, typically pounds per hour.	

Control Inputs		
N1_RPM	Type	Default Value
	float32	1.0
	Description: Provides control for the first stage of blade RPM.	
N2_RPM	Type	Default Value
	float32	1.0
	Description: Provides control for the second stage of blade RPM.	
Output-Gain	Type	Default Value
	float32	1.0
	Description: Applies the overall composite gain control.	

Internal Parameters		
Altitude	Type	Default Value
	function	n/a
	Description: The sound driven by Altitude input manipulated by Gain_Table to create the altitude effects on sound.	
Combustion_Fuelflow	Type	Default Value
	function	n/a
	Description: The sound driven by FuelFlow manipulated by Freq_Table and Gain_Table to create the combustion sound.	

Internal Parameters		
ExhaustRoar_Fuelflow	Type	Default Value
	function	n/a
	Description: The sound driven by Fuelflow manipulated by Freq_Table and Gain_Table to create the engine exhaust roar sound.	
IntakeHiss_Airspeed	Type	Default Value
	function	n/a
	Description: The sound driven by airspeed manipulated by Freq_Table and Gain_Table to create the intake hiss sound.	
IntakeHiss_RPM	Type	Default Value
	function	n/a
	Description: The sound driven by RPM manipulated by Freq_Table and Gain_Table to create the intake hiss sound.	
RotatingMachinery_RPM	Type	Default Value
	function	n/a
	Description: The sound driven by RPM manipulated by Freq_Table and Gain_Table to create the engine rotating machinery sound.	
TurboFan1_RPM	Type	Default Value
	function	n/a
	Description: The sound driven by RPM manipulated by Freq_Table and Gain_Table to create the engine fan sound.	

Internal Parameters		
TurboFan2_RPM	Type	Default Value
	function	n/a
	Description: The sound driven by RPM manipulated by Freq_Table and Gain_Table to create the engine fan sound.	
Whine_Frequencies	Type	Default Value
	function	n/a
	Description: Driven by RPM and used to drive up to six whine frequencies.	
Whine_Gain	Type	Default Value
	function	n/a
	Description: Driven by RPM and used to drive the amplitude of up to six whines.	

7.4. MultiFilter

Summary: Expanded version of the Audio/Filter component. The MultiFilter has a variety of basic filter types and the ability to specify the filter order.

Description: A signal is filtered by one of the basic filter types described below.

Off: No filtering occurs. OutGain affects OutSignal amplitude.

LowPass: Basic LowPass filter; frequency content below the specified frequency is passed through and content above is attenuated.

HighPass: Basic HighPass filter; frequency content below the specified frequency is attenuated and content above is passed through.

BandPass: Basic BandPass filter; frequency content ranging from StartBand to EndBand is passed through, content outside of this range is attenuated.

BandPassUnityGain: Same as the BandPass filter except that the amplitude of the BandPassFilter is scaled by 1/QFactor resulting in a 0 dB peak gain, whereas the normal BandPassFilter has a peak gain of Q.

Notch: Basic Notch filter; frequency content at the specified frequency is attenuated.

AllPass: Basic AllPass filter; no frequency is attenuated.

PeakingEQ: Basic Peak filter; frequency content near the specified frequency is boosted or attenuated by the Gain_dB control.

LowShelf: Basic LowShelf filter; frequency content below the specified frequency is boosted or attenuated by the Gain_dB control.

HighShelf: Basic HighShelf filter; frequency content above the specified frequency is boosted or attenuated by the Gain_dB control.

If the filter is not 'Enabled' the incoming InSignal is passed through unprocessed but potentially modified by the OutGain control. Ensure OutGain results in 1.0 and Enable or FilterType results in False or Off, respectively, for a true bypass of the component.

Audio Inputs		
InSignal	Type	Default Value
	audio	n/a
	Description: The signal to be filtered. This input should be linked into the component from somewhere else in the model.	

Control Inputs		
Enable	Type	Default Value
	boolean	True
	Description: This control determines if the signal is filtered. If this control results in False the signal is not filtered but the OutGain control is still applied. Modifier: XOR Modifier_default: False	
Frequency	Type	Default Value
	float32	1.0
	Description: This control specifies frequency in Hertz. For the BandPass, BandPassUnityGain, and PeakingEQ filters this control specifies the center frequency of the passband. For the Notch filter this control specifies the center frequency of the stopband. For the LowPass and HighPass filters this control specifies the corner (-3 dB) frequency. For the LowShelf and HighShelf filters this control specifies the midpoint frequency. The range of this control is [20, 24000). Modifier: Multiply Modifier_default: 500	

Control Inputs		
Gain_dB	Type	Default Value
	float32	0.0
	Description: This control specifies a gain value in dB. For Peak-ingEQ, HighShelf, and LowShelf filters this control specifies the gain applied to the passband. The range of this control is [-50,50]. Modifier: Plus Modifier_default: 0.0	
OutGain_dB	Type	Default Value
	float32	0.0
	Description: This control specifies gain in a linear scale. For all filters this control specifies the gain applied to the post-filtered signal. This control takes effect even if Enable results in False and if FilterType is 'Off'. The range of this control is [0, 316.2277] and a setting of 1.0 corresponds to a gain of 0 dB. Modifier: Multiply Modifier_default: 1.0	

Control Inputs		
QFactor	Type	Default Value
	float32	1.0
	Description: The filter quality factor. This control specifies the bandwidth for BandPass, BandPassUnityGain, Notch, and Peak-ingEQ filters; the higher this value, the smaller the bandwidth. The StartBand and EndBand variables adjust according to the QFactor and are not adjustable in the display. For HighPass, LowPass, High-Shelf, and LowShelf filters this control specifies the roll-off / corner frequency gain; the higher this value the steeper the roll-off and the higher the gain at the corner frequency. The range of this control is (0, 16) and setting this control to a value of 0 or lower results in the control defaulting to 0.707107. Modifier: Multiply Modifier_default: 0.7071.	

Audio Outputs		
OutSignal	Type	Default Value
	audio	n/a
	Description: The signal after it is filtered.	

Internal Parameters		
EndBand	Type	Default Value
	float32	965.4772
	Description: Specifies the end of the bandwidth affected by the filter settings. Not adjustable. For BandPass, BandPassUnityGain, and Notch filters this specifies the right -3 dB corner frequency in hertz. For PeakingEQ filters this specifies the midpoint frequency in hertz.	
FilterType	Type	Default Value
	filter_type3	LowPass
	Description: FilterType determines the type of filter applied to the input signal. If no filtering is desired, FilterType should be set to Off.	
Order	Type	Default Value
	filter_order	2nd_order_12dB_octave
	Description: FilterOrder determines the order of the filter applied to the input signal. 2 nd , 4 th , and 6 th orders are available, corresponding to 12, 24, and 36 dB per octave roll-off in LowPass, HighPass, LowShelf, and HighShelf filters and 6, 12, and 18 dB per octave roll-off in BandPass, BandPassUnityGain, Notch and PeakingEQ filters (assuming QFactor is fixed at 0.7071). Generally, the higher this control the steeper the roll-offs will be.	
StartBand	Type	Default Value
	float32	258.9393
	Description: Specifies the start of the bandwidth affected by the filter settings. Not adjustable. For BandPass, BandPassUnityGain, and Notch filters this specifies the left -3 dB corner frequency in hertz. For PeakingEQ filters this specifies the midpoint frequency in hertz.	

7.5. Prop Rotor

Summary: The Prop Rotor component provides the composite sound for a rotating helicopter blade.

Description: The Prop Rotor component includes the three principal sources of noise; air noise from the movement of air over the blades, force noise from the impact of the blade with the air medium, and thickness of noise due to the dual edge sound sources on a blade.

The overall sound can be tuned based upon blade parameters such as radius and blade count, with overall gain control based on both RPM and blade angle.

Control Inputs		
Angle	Type	Default Value
	float32	1.0
	Description: Provides the blade angle value.	
ForceGain	Type	Default Value
	float32	1.0
	Description: The gain for the principle component of the blade sound due to the air force on the blade. If a function connection is not made then this gain equals the scale factor, or it is the scale factor times the result of the selected function RPM and Blade Angle. Connect a table function for controlling the force gain based upon RPM and Blade Angle.	
Lagfilter	Type	Default Value
	float32	1.0
	Description: This provides a limiting filter used for fade-in and fade-out effects.	

Control Inputs		
NoiseGain	Type	Default Value
	float32	1.0
	Description: The gain for the air noise component of the blade sound. Connect to a table function for controlling the noise gains based upon RPM and Blade Angle. If the function connection is not made then the noise gain equals the scale factor, or it is the scale factor times the result of the selected function of RPM and Blade Angle.	
Output-Gain	Type	Default Value
	float32	1.0
	Description: Amplitude gain of prop/rotor composite sound source. If the gain connection is blank then the gain scale factor is used as the gain value, or the gain is the scale factor times the output result of the control component.	
Radius	Type	Default Value
	float32	1.0
	Description: This constant is used to scale tip mach speed based on blade radius (in meters).	
RPM	Type	Default Value
	float32	1.0
	Description: The frequency (in Revolutions per Minute) of the blade shaft.	
Thickness-Gain	Type	Default Value
	float32	1.0
	Description: Connect a table function for controlling the thickness gain based upon RPM and Blade Angle.	

Control Inputs		
Qfactor	Type	Default Value
	float32	1.0
	Description: Quality factor for the air noise filter. If the Q factor connection is blank then the Q scale factor is used as the Q value, or the Q is the scale factor times the output result of the control object.	

Audio Input		
InSignal	Type	Default Value
	audio	
	Description: Audio stream input links. All of the incoming audio is mixed in this component. The audio stream is used as an input to the filter. The input signal comes from another component in the model.	

Audio Output		
OutputSignal	Type	Default Value
	audio	true
	Description: Audio output stream from the PropRotor component.	

Internal Parameters		
Filter-FreqScale	Type	Default Value
	float32	4.0
	Description: Scale factor for noise filter roll-off frequency.	
FilterType	Type	Default Value
	filter_type2	BandPassQ
	Description: Selects a two pole filter type from: Lowpass, Bandpass, Highpass, LowpassQ, BandpassQ, and HighpassQ. The latter three are amplitude adjusted such that the filter has unity gain at the roll-off frequency, and maintains this gain as the quality factor is increased. The bandpass filters have the lowpass and highpass poles at the same roll-off frequency.	
Machlimit	Type	Default Value
	float32	.950
	Description: A limit for the calculated tip mach speed. This keeps the maximum mach speed to a predetermined maximum. Usually between 0.95 and 0.99, depending how dominant the thickness noise is required. The sound model is not accurate above 0.99 since supersonic effects start to dominate the sound spectrum.	
Numberof-Blades	Type	Default Value
	uint8	6
	Description: Number of blades on the shaft.	

7.6. Vibration Capture

Summary: The Vibration Capture component records an input signal specifically for use with ASTi's Vibration Analysis tool kit.

Description: This component is a simplified version of the Record Replay component. The capture length defaults to 30 seconds and continuous recording is not allowed. The Record parameter controls the recording, when True recording begins.

The captured file will result in a TSR file with 32 bit float values recorded at 48kHz with a name based on the GroupID and Sound Index. The file is written to the hard disk under /var/local/asti/recordreplay.

The component will always start recording at the beginning of the file and it cannot start in the middle of the file. The recording always overwrites the file from the beginning.

Audio Input		
SignalIn	Type	Default Value
	audio	n/a
	Description: Audio linked into this position is recorded and written to the record file on the hard disk.	

Control Inputs		
Gain	Type	Default Value
	float32	1.0.
	Description: Amplitude gain control for the capture file, which can be set via the host or internally.	

Control Inputs		
Sound Index	Type	Default Value
	uint16	0
	Description: Use to identify the file. Each file is organized as part of a group with a specific Index that is used to name the .tsr file. The file is written to the hard disk under /var/local/asti/recordreplay.	

Control Outputs		
FileMode	Type	Default Value
	player_state	Stopped
	Description: Displays the recording state of the file. Possible states are STOPPED and RECORDING.	
Record	Type	Default Value
	boolean	false
	Description: When True the component starts recording.	

Internal Parameter		
Length	Type	Default Value
	uint32	30
	Description: Set to the number of seconds to record vibration at a 48kHz sample rate. A value of 30 is the expected length for the Vibration Analysis.	

8.0. Highway Service

Summary: The highway service provides audio distribution for aural cues from components in ACE to the outside world. The need for a specialized service to handle aural cue audio is driven by the fact that aural cues are generated by multiple sources whose output is typically sent to multiple channels (e.g. left speaker, right speaker). The highway service handles both the mixing of audio from multiple sources and the routing of the composite audio to multiple channels.

The highway service supports 12 independent output channels. The service receives audio from a set of Balancer components. Each Balancer component sends a stream of audio to the service with applied gains for the audio on each of the 12 highways. The service mixes the audio sources for a given highway to create a composite audio stream for each highway. The service applies highway gains that it receives from a HighwayGain component. This set of gains includes an overall output gain that affects all highways and individual highway gains that affect all audio on a given highway. The service sends the composite audio to a HighwayOutput component, which is tuned to one and only one highway. The highway service supports input connections from only the Balancer and HighwayGain components and output connections to only the HighwayOutput component.

As with all service components, the user does not manually instigate the creation of the service object. Instead, this is done automatically on demand by the loader when it detects that a component has a highway service port connection. Only one highway service component will be loaded in a model based on the first found need for a service of this type.

Description: The Highway Service component receives an audio input and an array of 12 gain values (one for each highway) from each balancer component. The highway service can connect to up to 1024 balancers. The connection from the balancer to the highway service is unidirectional for audio data. The balancer is always the audio source while the highway service is always the audio sink.

For each balancer connection, the highway service effectively makes 12 copies of the audio and applies a different gain value to each copy. The gain values come from the gain value array provided by the balancer. The output from this step is a set of 12 audio streams (one for each highway) per balancer.

The highway service mixes the audio from the various balancers for a given highway into a final audio output for that highway.

The highway service also receives an input connection from the HighwayGain component, which provides an overall output gain and twelve individual highway gains. The highway service applies the gains it receives from the HighwayGain component to the composite audio on each highway. Each model may contain only

one HighwayGain component. The model loader should throw an error if a user attempts to add a second HighwayGain component. If there is no HighwayGain component in the model, the highway service should use a value of 1.0 for these gains.

The output connection from the highway service goes to a HighwayOutput component. This output connection is unidirectional for audio data. The highway service is always the audio source while the HighwayOutput is always the audio sink. Multiple HighwayOutput components may point to the same highway channel.

The service must also support the flow through of Active_TX flags to indicate the state of audio activity on a highway. The Active_TX flag should be set only if there is audio present and a non-zero gain value at a given point in the path.

The Highway Service tool will also support diagnostic facilities to allow the user to view the use of a highway within the model and the current highway gain settings. This will scan the model for all HighwayOutputs the HighwayGain and extract the information into a table-like view as demonstrated in the following example:

Handle view – this will display all HighwayOutputs including their paths that are connected to each handle as well as the gains associated with each highway (Note: This is just an example to show the type of information to be displayed.):

Overall Output Gain = 1.0

Left_Speaker (Gain = 0.5) --->

Aural_Cues / Left_Speaker_Sounds > HighwayOutput_A

Aural_Cues / Left_Speaker_Sounds > HighwayOutput_B

Seat_Shaker (Gain = 0.8) --->

Some_Folder1 / Some_Folder2 / Vibration_Output > HighwayOutput_C

8.1. AuralCue

Summary: Inputs audio into the Highways Service and associates the audio with a name or “cue ID.”

Description: The AuralCue component is the first step when using the Highways Service. First audio is linked to the AudioFeed component and the cue ID is created and selected. Then this audio is mixed in a 3D or non-3D way using the other Feeders components.

Audio Input		
InSignal	Type	Default Value
	audio	n/a
	Description: The audio linked into this input.	

Control Input		
Gain	Type	Default Value
	float32	1.0
	Description: A gain applied to the input audio streams.	

Internal Parameters		
Cue	Type	Default Value
	ID	unassigned
	Description: A name given to this audio source. For example, add and select a cue called Engine1 and route all engine audio into this AuralCue component. Note: Users should generate meaningful names for better readability in the model. Reference this cue name in a corresponding components.	

8.2. AuralCuePosn

Summary: Adds the selected sound to the soundfield at the specified (X, Y, Z) position.

Description: For example, in an aircraft model, you might have an AuralCuePos component with a Cue ID called “Left Engine” positioned at X= -3.0, Y= 3.0, Z= 0.0. This means that the audio fed into this Cue ID is positioned to the back-left of the reference point.

Once a cue is added to the soundfield with this component, add either a Highway>SpeakerOutput or an AudioIO>SpeakerOut component to get audio from the soundfield.

Internal Parameters		
CueID	Type	Default Value
	ID	unassigned
	Description: Selects the sound for positioning.	

Control Input		
X	Type	Default Value
	float64	0.0
	Description: The X coordinate relative to the reference point. Positive X is in front and negative X is behind the reference point.	
Y	Type	Default Value
	float64	0.0
	Description: The Y coordinate relative to the reference point. Positive Y is to the left and negative Y is to the right of the reference point.	
Z	Type	Default Value
	float64	0.0
	Description: The Z coordinate relative to the reference point. Positive Z is above and negative Z is below the reference point.	

8.3. SpeakerOutput

Summary: Retrieves audio from the Highways soundfield that is intended for a speaker at the specified X, Y, Z position.

Description: SpeakerOutput creates a speaker in the Highways Service with the specified X, Y, and Z coordinates relative to the soundfield reference point, extracting the audio from the service for the speaker. The X, Y, and Z position should correspond to the speaker's physical location. Typically, the AudioOut is linked directly to an AmpOut component's audio out.

For example, if using an eight channel crown amp to drive an 8 speaker setup, the model should contain 8 SpeakerOuts linked to 8 AmpOut components.

Audio Output		
AudioOut	Type	Default Value
	audio	n/a
	Description: Audio for this speaker.	

Internal Parameter		
Position	Type	Default Value
	worldposition_ geocentric	(0,0,0)
	Description: The X, Y, and Z position of the speaker relative to the reference point. The coordinate system is the same as the one used in the Highway > AuralCuePos component (+X is forward, +Y is to the left, and +Z is up).	
Source-Speaker	Type	Default Value
	id	unassigned
	Description: Selects the speaker for sound positioning.	

9.0. Highway 3D Service

The Highways 3D Service is a collection of components used for mixing and routing audio to hardware output channels. The service provides two approaches for audio mixing - Gain-Mixing and 3D Soundfield Reconstruction - making it useful for a range of applications, from routing communications audio to operator headsets to immersing listeners in a 3D audio environment.

Gain-Mixing

Gain-Mixing is the simpler approach and represents the more traditional way of mixing audio. Sounds are sent to a set of output “highways” using a set of gains that are driven by the host or calculated in the model. Gain-Mixing using the Highways 3D Service allows the developer to easily route audio from simulation models to hardware models, making it an attractive alternative to using connectors and busses.

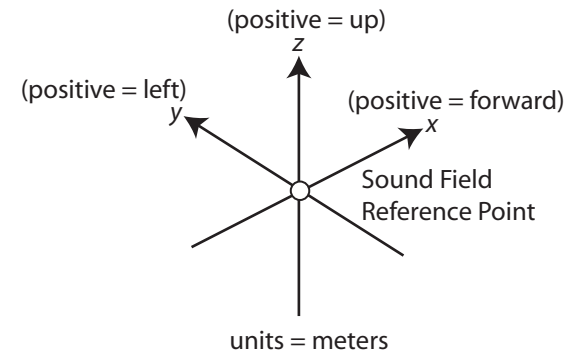
Soundfield Reconstruction

The 3D Soundfield Reconstruction (SFRC) is a sophisticated method of processing sounds to make them appear to come from somewhere in 3D space around the listener(s). Sounds are assigned (x, y, z) positions relative to the listening or “reference” point. The Highways 3D Service then filters and mixes each sound based on its position in order to encode the sound into a virtual 3D sound environment or “Soundfield”.

Once all the sounds are encoded, the Soundfield is decoded for playback in stereo headphones or an array of speakers. If decoding to headphones, two output streams are output for the left and right earphones. The head's orientation is used in this calculation if it is available from a head-tracking system. If decoding to a speaker array, the service outputs one audio stream for each speaker, taking into account the speaker's (x, y, z) position.

All positions used in the service must be specified in meters relative to the listening or “reference” position. The reference position should be the central listening point of the simulator, such as the midpoint between the pilot's and co-pilot's heads. This ensures the service will render the sounds in a way that sounds best to all listeners.

Use the following coordinate system when calculating the relative positions of sounds and speakers:



The 3D positioning algorithm used by the service calculates a contribution from each speaker for each sound, even if the speaker is not precisely in the right direction. This results in realistic and immersive reproduction of many types of sounds.

If a sound is to output only through one speaker, use gain-mixing or connectors and busses instead of SFRC.

How to use the Highways 3D Service

Using the Highways 3D Service is a three-step process that includes entering audio into the service, determining how it is mixed or processed, and extracting audio for the outputs from the service.

The Highways 3D Service components include:

For input into the service:

Audio > AudioFeed

For controlling gain-mixing or 3D positioning of a sound:

Feeders > Balancer1, 4, 8, 16

Feeders > AuralCuePos

For extracting audio from the service:

Audio IO > HighwayOut

Audio IO > SpeakerOut

Audio IO > Headphone3dOut

Add an AudioFeed component in a simulation model to input a sound into the service. Add more AudioFeed components for other sounds if they need mixing or independent positioning. Create and select new sound names or “cue IDs” to identify the sound in the service.

Add a BalancerN component in a Feeder model to gain-mix a sound to the output highways. Select the cue ID of the sound used for mixing and then select or create the highways. The number of output highways the sound is mixed to is determined by the size of the Balancer. Link the gains from the host or elsewhere in the load if desired, or just set static gains by hand.

Add an AuralCuePos component in a Feeder model to add a sound to the 3D Soundfield at a specified position. Select the cue ID of the sound that is then added to the Soundfield. Link to or set the (x, y, z) position inputs to determine the location of the sound in the Soundfield.

Add a HighwayOut component in a hardware model to extract audio from one of the output highways. Select the highway and then link the audio output of the component to a hardware output. Only sounds put onto the highways using Balancer components are retrieved by this component, not sounds that were positioned in the Soundfield using the AuralCuePos component.

Add SpeakerOut components in a hardware model for each speaker in the simulator. Enter the (x, y, z) position of the speaker and then link the audio output

to a hardware output. The service will decode the Soundfield to this speaker based on the specified position.

Add a Headphone3DOut component in a hardware model for operators that should have 3D audio in their stereo headphones. If head tracker information is available from the host, link it to the head azimuth and elevation inputs. Link the left and right audio outputs of the component directly to the output channels for the left and right earcups of the operator's headphones. Avoid adding other components between the Headphone3DOut and hardware components. The presence of other components on these audio streams can alter the timing and distort the apparent 3D position of the sound.

Note: HighwayOut and SpeakerOut components can be added to auto-generated hardware models using the Channel helper. Notice the “Highway”, “Speaker”, and “Speaker Position” options when creating ACE-RIU, ACU, and Amp channels in the Channels helper window. Enter a highway name to have a HighwayOut component created, the highway selected, and the audio linked to the channel's audio out. Enter a speaker name and position to have a SpeakerOut component created at that position with its audio linked to the channel's audio out.

The Gain-Mixing figure below shows an example of using the Highways 3D service for Gain-Mixing warning tones to a set of output highways.

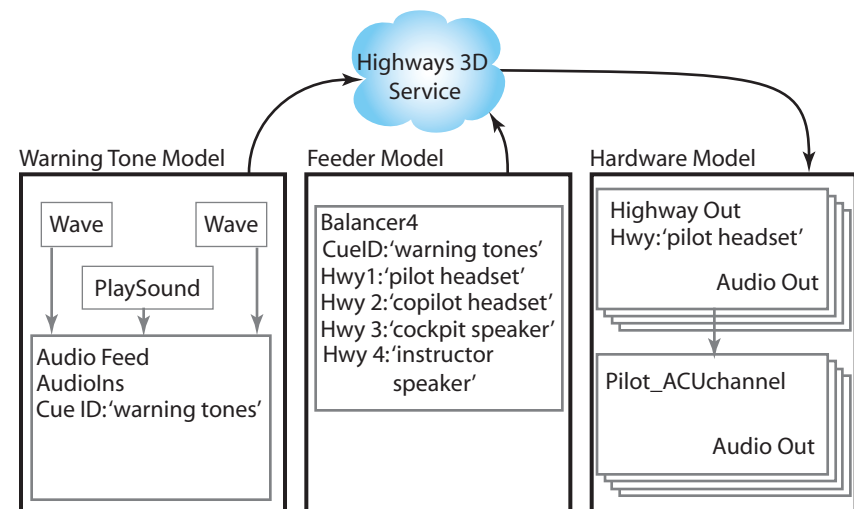


Figure 12: Gain-Mixing

The Soundfield Reconstruction figure below shows an example of using the Highways 3D service for position weapons sounds in 3D and rendering the Soundfield to both a speaker array and a stereo headset:

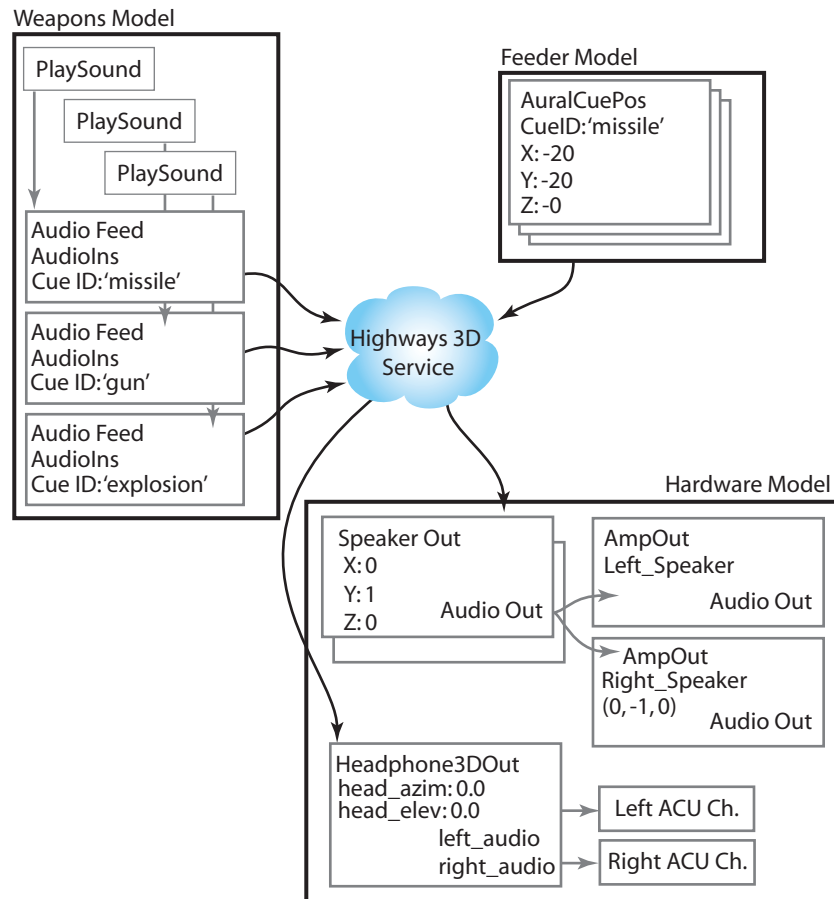


Figure 13: Soundfield Reconstruction

See the ACE Studio Technical User Guide (DOC-01-TELAS-UG-4) "Highway 3D Service" section for more information.

9.1. Audio > Audio Feed

Summary: Inputs audio into the Highways 3D Service and associates the audio with a name or “cue ID.”

Description: The AudioFeed component is the first step when using the Highways 3D Service. First audio is linked to the AudioFeed component and the cue ID is created and selected. Then this audio is mixed in a 3D or non-3D way using the other Feeders components.

Audio Inputs		
InSignal	Type	Default Value
	audio [64]	n/a
	Description: An array of audio inputs. Up to 64 links can link into this input. All of the incoming audio is mixed in this component.	
InSignals	Type	Default Value
	audio	n/a
	Description: Provides a view of all the mixed audio inputs.	

Control Inputs		
Gain	Type	Default Value
	float	1.0
	Description: A gain applied to the input audio streams.	

Internal Parameters		
Cue	Type	Default Value
	ID	unassigned
	Description: A name given to this audio source. For example, add and select a cue called Engine1 and route all engine audio into this AudioFeed component. Note: Users should generate meaningful names for better readability in the model. Reference this cue name in a corresponding Balancer and AuralCue-Pos component.	

9.2. Feeders > Aural Cue Posn

Summary: Adds the selected sound to the soundfield at the specified (X, Y, Z) position.

Note: This component can only be added to Feeder models.

Description: For example, in an aircraft model, you might have an AuralCuePos component with a Cue ID called “Left Engine” positioned at X= -3.0, Y= 3.0, Z= 0.0. This means that the audio fed into this Cue ID (with the AudioFeed component) is positioned to the back-left of the reference point.

Once a cue is added to the soundfield with this component, add either an AudioIO > Headphone3DOut or AudioIO>SpeakerOut component to get audio from the soundfield.

Control Inputs		
X	Type	Default Value
	float	0.0
	Description: The X coordinate relative to the reference point. Positive X is in front and negative X is in back of the reference point.	
Y	Type	Default Value
	float	0.0
	Description: The Y coordinate relative to the reference point. Positive Y is left and negative Y is right of the reference point.	
Z	Type	Default Value
	float	0.0
	Description: The Z coordinate relative to the reference point. Positive Z is above and negative Z is below of the reference point.	

Internal Parameters		
CueID	Type	Default Value
	ID	unassigned
	Description: Selects the sound for positioning.	

9.3. Feeders > Balancer1, 4,8,16

Summary: Mixes the selected sound to the selected highways with a set of gains. Balancer 1 mixes to 1 Highway, Balancer 4 to 4 Highways, etc.

Note: This component can only be used in Feeder models.

Description: After inputting the audio into the Highway 3D service using the AudioFeed component, Balancers are used to mix the audio to output highways. Balancers provide simple gain mixing to the highways - no 3D calculations are involved. If positioning in 3D is desired, use the Feeders> AuralCuePos component.

For example, use Balancer4 to mix a tone to four different highways where the audio from each highway is routed to an operator's headset.

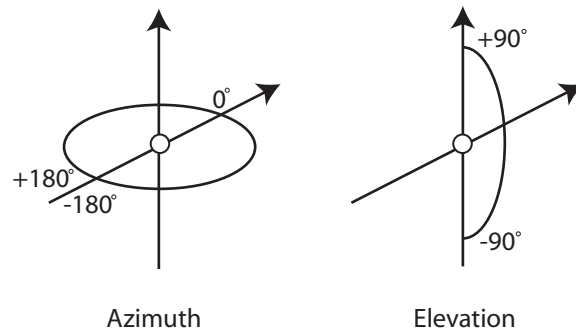
Control Inputs		
Gain	Type	Default Value
	float	1.0
	Description: The gain applied to the cue's audio before it is mixed to the highways.	
Gain1-16	Type	Default Value
	float	1.0
	Description: An additional gain applied when mixing the cue's audio to each highway. Gain1 affects mixing onto the first selected highway, Gain2 affects mixing onto the second selected highway, etc.	

Internal Parameters		
Cue	Type	Default Value
	ID	unassigned
	Description: Selects the sound for mixing.	
Highways	Type	Default Value
	ID [16]	unassigned
	Description: Selects the highways for mixing. The number of highways varies based on which Balancer is used (1, 4, 8, or 16).	

9.4. AudioIO > Headphone3DOut

Summary: The Headphone3DOut component retrieves the soundfield from the service and filters it to reproduce the 3D environment in a pair of audio outputs for a stereo headset.

Description: Use this component to generate 3D audio in the headphones of an operator. The operator's head is assumed to be located at the soundfield reference point (0, 0, 0). Head azimuth and head elevation are typically driven from the host based on values from a head tracking system. These values specify the head's orientation relative to straight forward and cause the soundfield to rotate accordingly.



Control Inputs		
Enable	Type	Default Value
	boolean	true
	Description: Enables and disables the 3D filtering. If disabled, the sound field is mixed in mono to the left and right headphone channels.	
HeadAzim	Type	Default Value
	floats	0.0
	Description: The orientation of the head in the horizontal plane, 90 degrees is directly to the left and -90 degrees is directly to the right.	

Control Inputs		
HeadElev	Type	Default Value
	float	0.0
	Description: The orientation of the head in the vertical plane, 90 degrees is directly straight up and -90 degrees is directly straight down.	

Audio Outputs		
LeftAudio	Type	Default Value
	audio	n/a
	Description: The audio intended for the left earcup of the headphones.	
RightAudio	Type	Default Value
	audio	n/a
	Description: The audio intended for the right earcup of the headphones.	

9.5. AudioIO > HighwayOut

Summary: Extracts the specified Highway's audio from the Highway 3D Service. The AudioOut typically is linked directly to a hardware (ACE-RIU, ACU, etc.) component's audio out.

Description: To create new highways as needed double-click on the Source Highway value and then add new busses in the service window. For example, add a highway for each speaker or headset. Name the highways based on the corresponding hardware that it is linked to, e.g. "Pilot Headset".

To put audio onto the highways, use the Audio> AudioFeed component and a Feeders>Balancer component. The Balancer takes audio from the AudioFeed and puts it onto one or more highways.

Control Input		
Gain	Type	Default Value
	float32	n/a
	Description: A gain applied to the Audio output.	

Audio Output		
AudioOut	Type	Default Value
	audio	n/a
	Description: The highway's audio.	

Internal Parameter		
Source Highway	Type	Default Value
	ID	unassigned
	Description: Specifies which highway to extract audio from.	

9.6. AudioIO > SpeakerOut

Summary: Retrieves audio from the Highways 3D soundfield that is intended for a speaker at the specified X, Y, Z position.

Description: SpeakerOut creates a speaker in the Highway 3D Service with the specified X, Y, and Z coordinates relative to the soundfield reference point, extracting the audio from the service for the speaker. The X, Y, and Z position should correspond to the speaker's physical location. Typically, the AudioOut is linked directly to an AmpOut component's audio out.

For example, if using an eight channel crown amp to drive an 8 speaker setup, the model should contain 8 SpeakerOuts linked to 8 AmpOut components.

Control Input		
Gain	Type	Default Value
	float32	n/a
	Description: A gain applied to the audio output.	

Audio Output		
AudioOut	Type	Default Value
	audio	n/a
	Description: Audio for this speaker.	

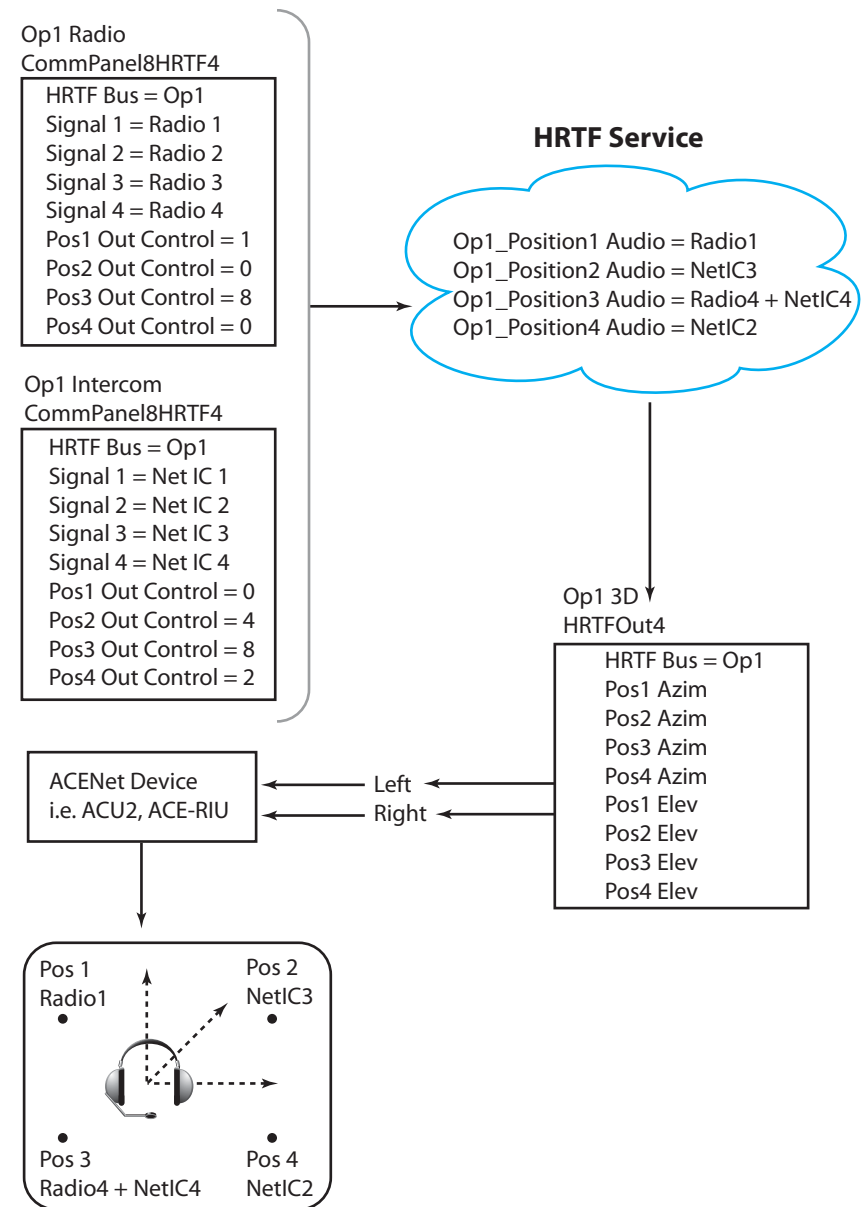
Internal Parameter		
Position	Type	Default Value
	worldposition_ geocentric	(0,0,0)
	Description: The X, Y, and Z position of the speaker relative to the reference point. The coordinate system is the same as the one used in the Feeders > AuralCuePos component (+X is forward, +Y is to the left, and +Z is up).	

10.0. HRTF Service

Description: The HRTF service provides 3D capability for headphone audio. Both environmental and communications audio can be mixed in 3D. Head-related-transfer functions (HRTFs) position audio streams at specified 3D (azimuth and elevation) positions.

The following example depicts a 3D communications configuration for a single operator. The operator has two CommPanels, one for radios and the other for intercoms. Since both CommPanels have the same HRTF bus selected (Op1), they will contribute audio to the same four audio positions. Behind the scenes, the HRTF service mixes audio from CommPanels with the same HRTF bus. Then the HRTFOut4 component applies 3D filters according to the specified azimuth and elevation positions. The HRTFOut4's left and right audio outputs connect to an ACENet channel to go to the operator's stereo headset.

Each additional 3D operator may have a nearly identical configuration, but must use a unique HRTF bus for independent audio and 3D positioning.



10.1. HRTFOut4

Summary: Outputs a 3D mix of audio for up to four sound positions.

Description: This component receives up to 4 audio streams from HRTF service components such as the CommPanel8HRTF4. Each stream is positioned at a specific azimuth and elevation in 3D space relative to the listener.

Valid azimuth values range is from -180 to 180 degrees. Valid elevation range is from -40 to 90 degrees. HRTF measurements are available in increments of a few degrees. The azimuth and elevation inputs will jump to the nearest position for which a filter is available. The resulting azimuth and elevation is then shown in the result column, for each control.

An azimuth and elevation position of (0,0) corresponds to directly in front of the listener. Positive azimuths position sounds to the right and negative azimuths position sounds to the left. An azimuth of 90 corresponds to a position in line with the left ear. An elevation of 90 degrees means straight up.

The HRTF uses time (i.e. delay) and intensity (i.e. gain) differences to synthesize how a sound or channel appears to come from a particular point in space. It is REQUIRED that the HRTFOut4 component Left and Right audio outputs directly connect to the ACENet channel(s) since that is the final output stage before the D/A conversion. For example, if the HRTF is using an ACU2 device, the HRTFOut4 component should directly link to the ACU2 component rather than going through a series of mixers or other components. This ensures that the HRTF timing and gains are maintained. Other audio sources (sidetone, etc.) can be mixed into the ACENet component as needed, given the ACENet component performs implicit mixing at the final output stage.

Control Inputs		
Enable	Type	Default Value
	boolean	True
	Description: Enables or disables the 3D filtering on all 4 audio positions. If False, the audio outputs are a mono mix of the 4 positions.	

Control Inputs		
Pos1Azim - Pos4Azim	Type	Default Value
	int32	0
	Description: Azimuth in degrees of audio position N (1-4).	
Pos1Elev - Pos4Elev	Type	Default Value
	int32	0
	Description: Elevation in degrees of audio position N (1-4).	
Pos1Gain - Pos4Gain	Type	Default Value
	float 32	1.0
	Description: Gain applied to position N (1-4) audio before mixing.	
OutGain	Type	Default Value
	float32	1.0
	Description: Gain applied to the left and right audio outputs, affecting all positions.	

Audio Outputs		
OutLeft	Type	Default Value
	audio	n/a
	Description: 3D audio output for the left ear. Warning: OutLeft must be directly connected to an ACENet component (i.e. ACU2, ACE-RIU, etc.). See component description for further details.	

Audio Outputs		
OutRight	Type	Default Value
	audio	n/a
	Description: 3D audio output for the right ear. Warning: OutRight must be directly connected to an ACENet component (i.e. ACU2, ACE-RIU, etc.). See component description for further details.	

Status		
Pos1Audio - Pos4Audio	Type	Default Value
	audio	n/a
	Description: Displays each position's audio stream prior to 3D mixing.	

Internal Parameters		
HRTFBus	Type	Default Value
	id	n/a
	Description: Selects the HRTF service bus of this component. Components sending audio to this component (like CommPanel8HRTF4) must have a matching bus.	

10.2. CommPanel8HRTF4

Summary: Operates the same as the CommPanel8, but outputs audio with the HRTF service for 3D positioned communications.

Description: The CommPanel8HRTF4, much like a standard CommPanel, transmits and receives on multiple intercoms or radios. The component's InSignal and SideSignal behavior is identical to those of the generic comm panel. However, instead of one OutSignal link, four output signals are generated and sent to the HRTF service for 3D mixing. The four signals correspond to four positions in 3D space.

This component should be paired with an Audio/HRTFOut4 component, which receives the four output signals and does the 3D filtering. To establish this connection, the components must have matching HRTF busses. The four PosNOutControl masks determine which intercom buses contribute receive audio to each 3D position.

Multiple CommPanel8HRTF4 components can share the same HRTF bus to contribute audio to the same set of four positions.

Audio Inputs		
InSignal	Type	Default Value
	audio	n/a
	Description: Input audio to be transmitted by the Comm Panel.	

Control Inputs		
InControl	Type	Default Value
	byte	255
	Description: Bitmask that selects the intercom buses to transmit the InSignal. For example, a value of 1 will transmit on Sig1, a value of 2 on Sig2, 4 on Sig3, and 255 on all buses.	

Control Inputs		
InGain	Type	Default Value
	float32	1.0
	Description: Scales the input audio.	
Pos1Out Control- Pos4Out Control	Type	Default Value
	byte	255
	Description: Bitmask that selects which buses will contribute their output signal to the position N (1-4) output.	
Power	Type	Default Value
	boolean	True
	Description: Controls the power of the comm panel. If False no audio is received or transmitted.	
PTT	Type	Default Value
	boolean	False
	Description: Controls comm panel audio transmission.	
Side Control	Type	Default Value
	byte	255
	Description: Bitmask that selects the intercom buses that will contribute to the SideSignal. For example, a value of 1 will receive from Sig1, a value of 2 from Sig2, 4 from Sig3, and 255 from all buses.	

Control Inputs		
Sidetone Gain	Type	Default Value
	float32	1.0
	Description: Scales the sidetone mix just before the SideSignal output.	
SideGain Control	Type	Default Value
	byte	255
	Description: Selects which sidetones are affected by the receive signal gains. When a bit is high, sidetone volume for the intercom bus is multiplied by the appropriate SigNRxGain. Also when bit is high the SideControl value for the intercom bus becomes the logical AND of SideControl and OutControl.	
Sidetone Local	Type	Default Value
	byte	255
	Description: Selects which sidetones are generated locally vs. remotely. If more than one comm panel is sharing the same intercom bus, this control determines if the sidetone is shared with the other panels.	
Sig1-8_RxGain	Type	Default Value
	float32	1.0
	Description: Receive volume for each intercom bus.	

Audio Output		
SideSignal	Type	Default Value
	audio	n/a
	Description: Sidetone audio generated by mixing all the received sidetones from the buses selected in the SideControl bitmask.	

Internal Parameters		
HRTFBus	Type	Default Value
	id	n/a
	Description: Selects the HRTF service bus. The HRTFOut4 component with a matching bus will receive this commpanel's four position audio streams.	
Sig1-8	Type	Default Value
	id	n/a
	Description: Selects the intercom bus handle.	

11.0. IOInterfaces

The following section details the IOInterfaces components and the objects within them.

The IOInterfaces components include:

- ACE_RIU_channel
- ACE_RIU_SerialByteOut
- ACUchannel
- ACU2channel
- ACU2_SerialByteOut
- AmpOut
- SerialPort
- VoisusChannel

11.1. ACE_RIU_channel

Summary: The ACE-RIU device provides remote digital-analog audio and I/O distribution between Targets and audio peripherals. The ACE-RIU channel component assigns audio inputs and outputs for models.

Description: The ACE_RIU_channel component connects the software model audio to one specified channel on the ACE-RIU device. It also handles the digital in and out from the device channel.

Up to 32 audio streams can be mixed together and linked to the ACE-RIU channel output. The component accepts a single audio in source.

The audio strength is adjusted through the audio in and out gains. The overall channel volume is adjusted through the volume input.

Two serial devices can be connected to one ACE-RIU device. The serial devices are associated with ACE-RIU channels A and C. When connecting to a state machine device such as an HHT or SINCGARS panel, only use channels A and C.

Audio Input		
Audio-Outs 0-31	Type	Default Value
	audio	
	Description: Mixes up to 32 audio streams before it is routed out of the ACE-RIU device.	

Control Inputs		
AudioOut-Gain	Type	Default Value
	float32	1.0
	Description: Provides the overall gain control for the AudioOut.	
AudioIn-Gain	Type	Default Value
	float32	1.0
	Description: Provides the overall gain control for the AudioIn.	

Control Inputs		
DigitalOut	Type	Default Value
	boolean	false
	Description: Digital Out allows the software to drive a digital output on the ACE-RIU. Each ACE-RIU channel has 1 digital output.	
Volume	Type	Default Value
	float32	1.0
	Description: Sets the main volume of the ACE-RIU channel.	

Audio Output		
AudioIn	Type	Default Value
	audio	
	Description: Audio routed in from the ACE-RIU device and sent to other components in the model.	

Control Output		
DigitalIn	Type	Default Value
	boolean	
	Description: Digital In allows the ACE-RIU channel to drive a digital input to the software. Each ACE-RIU channel has 1 digital in.	

Internal Parameters		
Channel	Type	Default Value
	riu_channel	n/a
	Description: Select the ACE-RIU channel.	
Identifier	Type	Default Value
	device_id	n/a
	Description: Select the name of the ACE-RIU device.	
StateMachine	Type	Default Value
	strings	
	Description: Defines variables related to a state machine device associated with the specified ACE-RIU channel such as an HHT or SINGARS panel. State machine devices can only connect to channels A and C. Name - string: Identifies state machine instance in the .ini file. EntityName - string: Identifies state machine .ini file. Type sme_type/none: Type of state machine device.	

11.2. ACE_RIU_SerialByteOut

Summary: This component allows the user to transmit up to 8 bytes of data from an ACE_RIU ACENet device using the serial port interface.

Description: This device transmits a range of bytes (0-8) from the ACENet device's serial port interface. The bytes are transmitted whenever the MsgSize control is updated and as long as it is not 0. The bytes are also transmitted whenever any of the Char numbers are updated and as long as the updated Char number is in the range of bytes transmitted as specified by MsgSize.

Control Inputs		
MsgSize	Type	Default Value
	uint32	0
Description: Specifies the number of bytes that are transmitted from the CharsOut array. The range for this control is [0,8]. When this control is changed the corresponding number of bytes are transmitted in order. Modifier: Plus Modifier_default: 0		
CharsOut	Type	Default Value
	uint8	0
Description: An array of 8 bytes (uint8). The resulting bytes in this array are transmitted in order from Char1 to the number specified in MsgSize. If MsgSize is 0 no bytes are transmitted. If MsgSize is not 0 and a byte is updated in the range [Char1, Char<MsgSize>] the whole range is transmitted. Modifier: Plus Modifier_default: 0		

Internal Parameters		
Identifier	Type	Default Value
	device_id	n/a
Description: The name of the ACE_RIU goes here.		
Channel	Type	Default Value
	acenet_channel	n/a
Description: Select the ACE_RIU channel to transmit serial data. Serial Port A requires channel A, Serial Port B requires channel C.		
BaudRate	Type	Default Value
	serial_baud_rate	Baudrate_4800
Description: Select the baudrate of the serial data.		

11.3. ACUchannel

Summary: The software configurable component for the ACU hardware device.

Description: The ACU channel connects the software model audio to a channel of the physical ACU device. The control outputs provide the PTT channel selection for channels 0-3. Adjust audio strength with the AudioOut and AudioIn gains. The volume controls the overall channel's volume.

Audio Input		
AudioOut0-31	Type	Default Value
	audio	n/a
	Description: Mixes up to 32 audio streams before it is routed out of the ACU device.	

Control Inputs		
AudioIn-Gain	Type	Default Value
	float32	1.0
	Description: Provides the overall gain control for the AudioIn.	
AudioOut-Gain	Type	Default Value
	float32	1.0
	Description: Provides the overall gain control for the AudioOut.	
DigitalOut	Type	Default Value
	boolean	false
	Description: The ACU digital output allows the software to drive a digital output for the ACU channel. There is one digital output per channel.	

Control Inputs		
Volume	Type	Default Value
	float32	1.0
	Description: Sets the main volume of the ACU channel.	

Audio Output		
AudioIn	Type	Default Value
	audio	false
	Description: Audio routed in from the ACU device on the specified channel and becomes available to other components in the model.	

Control Outputs		
PTT	Type	Default Value
	boolean	false
	Description: When True the PTT is latched on for transmission.	
PTTselect	Type	Default Value
	uint8	255
	Description: Digital in for the PTT. Ranges 1-255	
PTT1	Type	Default Value
	boolean	false
	Description: When True the PTT is latched on for transmission.	

Control Outputs		
PTTselect1	Type	Default Value
	uint8	255
	Description: Digital in for the PTT. Ranges 1-255	
PTT2	Type	Default Value
	boolean	false
	Description: When True the PTT is latched on for transmission.	
PTTselect2	Type	Default Value
	uint8	255
	Description: Digital in for the PTT. Ranges 1-255	
PTT3	Type	Default Value
	boolean	false
	Description: When True the PTT is latched on for transmission.	
PTTselect3	Type	Default Value
	uint8	255
	Description: Digital in for the PTT. Ranges 1-255	

Control Outputs		
DigitalIn1	Type	Default Value
	uint8	0
	Description: Digital In allows the ACU to drive a digital input to the software and allows direct connection of PTT. Each ACU has 3 digital inputs per channel.	
DigitalIn2	Type	Default Value
	uint8	0
	Description: Digital In allows the ACU to drive a digital input to the software and allows direct connection of PTT. Each ACU has 3 digital inputs per channel.	
DigitalIn3	Type	Default Value
	uint8	0
	Description: Digital In allows the ACU to drive a digital input to the software and allows direct connection of PTT. Each ACU has 3 digital inputs per channel.	

Internal Parameters		
Channel	Type	Default Value
	acenet_channel	n/a
	Description: Select the ACU channel.	
Identifier	Type	Default Value
	device_id	n/a
	Description: Select the name of the ACU device.	
StateMa- chine	Type	Default Value
		n/a
	Description: Defines variables related to a state machine device associated with the specified ACU channel such as an HHT or SIN-CGARS panel. State machine devices can only connect to channels A and C. Name - string: Identifies state machine instance in the INI file. EntityName - string: Identifies state machine INI file. Type sme_type/none: Type of state machine device.	

11.4. ACU2channel

Summary: The software configurable component for the ACU2 hardware device.

Description: The ACU2channel connects the software model audio to a channel of the physical ACU2 device. The control outputs provide the PTT channel selection for channels 0-3. Adjust audio strength with the AudioOut and AudioIn gains. The volume controls the overall channel's volume. The ACU2 has 4 stereo audio inputs/outputs that provide support for multiple mono/stereo operators and audio equipment.

Audio Input		
AudioOut0-31	Type	Default Value
	audio	n/a
	Description: Mono audio output of the ACU2channel. Mixes up to 32 audio streams.	
AudioOut-sLeft	Type	Default Value
	audio	n/a
	Description: Audio that is routed out the left stereo channel. This audio is also mixed with the AudioOut audio streams. Mixes up to 32 audio streams.	
AudioOut-sRight	Type	Default Value
	audio	n/a
	Description: Audio that is routed out the right stereo channel. This audio is also mixed with the AudioOut audio streams. Mixes up to 32 audio streams	

Control Input		
AudioOut-Gain	Type	Default Value
	float32	1.0
	Description: The gain control for the AudioOut.	
AudioOut-GainL	Type	Default Value
	float32	1.0
	Description: The gain control for the AudioOutsLeft audio output.	
AudioOut-GainR	Type	Default Value
	float32	1.0
	Description: The gain control for the AudioOutsRight audio output.	
AudioIn-Gain	Type	Default Value
	float32	1.0
	Description: The gain control for the AudioIn.	
DigitalOut	Type	Default Value
	float32	1.0
	Description: The ACU2 digital output allows the software to drive a digital output for the ACU2channel. There is one digital output per channel.	
Volume	Type	Default Value
	float32	1.0
	Description: Sets the main volume of the ACU2channel outputs.	

Audio Output		
AudioIn	Type	Default Value
	audio	n/a
	Description: Audio routed in from the ACU2 device on the specified channel and becomes available to other components in the model.	

Control Output		
AnalogIn1	Type	Default Value
	uint8	0
	Description: Corresponds to DigitalIn1 with a range of 1-255. Digital in for the PTT. Used for 4-channel selector PTT.	
AnalogIn2	Type	Default Value
	uint8	0
	Description: Corresponds to DigitalIn2 with a range of 1-255. Used for 4-channel selector PTT.	
AnalogIn3	Type	Default Value
	uint8	0
	Description: Corresponds to DigitalIn3 with a range of 1-255. Digital in for the PTT. Used for 4-channel selector PTT.	
DigitalIn1	Type	Default Value
	boolean	False
	Description: Digital In allows the ACU2 to drive a digital input to the software and allows direct connection of PTT. Each ACU2 has 3 digital inputs per channel.	

Control Output		
DigitalIn2	Type	Default Value
	boolean	False
	Description: Digital In allows the ACU2 to drive a digital input to the software and allows direct connection of PTT. Each ACU2 has 3 digital inputs per channel.	
DigitalIn3	Type	Default Value
	boolean	False
	Description: Digital In allows the ACU2 to drive a digital input to the software and allows direct connection of PTT. Each ACU2 has 3 digital inputs per channel.	
PTT	Type	Default Value
	boolean	False
	Description: When True, the PTT is latched on for transmission.	
PTTselect	Type	Default Value
	uint8	255
	Description: Digital in for the PTT. Ranges 1-255	

Internal Parameters		
Channel	Type	Default Value
	acenet_channel	n/a
	Description: Select the ACU2 channel A-D.	
Identifier	Type	Default Value
	device_id	n/a
	Description: Select the name of the ACU2 device.	
StateMa- chine	Type	Default Value
		n/a
	Description: Defines variables related to a state machine device associated with the specified ACU2 channel such as an HHT or SINCGARS panel. State machine devices can only connect to channels A and C. Name - string: Identifies state machine instance in the .ini file. EntityName - string: Identifies state machine .ini file. Type sme_type/none: Type of state machine device.	

11.5. ACU2_SerialByteOut

Summary: This component allows the user to transmit up to 8 bytes of data from an ACU2 ACENet device using the serial port interface.

Description: This device transmits a range of bytes (0-8) from the ACENet device's serial port interface. The bytes are transmitted whenever the MsgSize control is updated and as long as it is not 0. The bytes are also transmitted whenever any of the Char numbers are updated and as long as the updated Char number is in the range of bytes transmitted as specified by MsgSize.

Control Inputs		
MsgSize	Type	Default Value
	uint32	0
Description: Specifies the number of bytes that are transmitted from the CharsOut array. The range for this control is [0,8]. When this control is changed the corresponding number of bytes are transmitted in order. Modifier: Plus Modifier_default: 0		
CharsOut	Type	Default Value
	uint8	0
Description: An array of 8 bytes (uint8). The resulting bytes in this array are transmitted in order from Char1 to the number specified in MsgSize. If MsgSize is 0 no bytes are transmitted. If MsgSize is not 0 and a byte is updated in the range [Char1, Char<MsgSize>] the whole range is transmitted. Modifier: Plus Modifier_default: 0		

Internal Parameters		
Identifier	Type	Default Value
	device_id	n/a
Description: The name of the ACU2 goes here.		
Channel	Type	Default Value
	acenet_channel	n/a
Description: Select the ACU2 channel to transmit serial data. Serial Port A requires channel A, Serial Port B requires channel C.		
BaudRate	Type	Default Value
	serial_baud_rate	Baudrate_4800
Description: Select the baudrate of the serial data.		
SerialSig-Type	Type	Default Value
	serial_signal_type	RS422
Description: Select the serial signal type.		

11.6. AmpOut

Summary: The software configurable component for the amplifier hardware devices.

Description: The AmpOut component drives up to 32 audio streams from within the model to a specified channel of the amplifier.

Audio Input		
AudioOut0-31	Type	Default Value
	Description: Mixes up to 32 audio streams before it is routed out of the amplifier channel.	

Control Inputs		
AudioOut-Gain	Type	Default Value
	float32	1.0
	Description: Provides the overall gain control for the AudioOut.	
Limiter Enable	Type	Default Value
	boolean	True
	Description: Turns the limiter stage on or off in the component. If True, the peak output levels of the component will not go above the LimThreshold.	
Limiter Release	Type	Default Value
	float32	1.0
	Description: The rate, in dB/sec, that determines how quickly the limiter responds to a drop in signal levels. Higher releases cause a faster increase in gain to ensure quiet signals are not reduced in volume.	

Control Inputs		
Limiter Threshold	Type	Default Value
	float32	1.0
	Description: The maximum peak level in dB allowed in the output signal. -12 corresponds to 0.25 linear, or 1/4 of a “full scale” signal.	
Volume	Type	Default Value
	float32	1.0
	Description: Sets the main volume of the amplifier.	

Internal Parameters		
Channel	Type	Default Value
	acenet_channel	n/a
	Description: Selects the amplifier channel.	
Identifier	Type	Default Value
	device_id	n/a
	Description: Select the name of the amplifier.	

11.7. SerialPort

Summary: This component is an internal ASTi component used to debug ACE-RIU channels.

11.8. VoisusChannel

Summary: The Voisus Channel component provides the audio communication and radio control for a remote client.

Description: This component is used with Comm Panels and multiple radio components to provide remote radio control. Use Voisus Channel to provide PTT, output, radio net selection, and volume controls for the remote client. This component accepts receive/transmit states and Comm Plan configuration for the remote client radios.

There are several settings per radio (1-16) that define how the remote operator controls the radio.

Audio Inputs		
AudioOut	Type	Default Value
	audio	n/a
	Description: Audio that is sent to the remote client.	
Sidetone	Type	Default Value
	audio	n/a
	Description: Audio that is sent to the remote client as the sidetone signal.	

Control Inputs		
Fill	Type	Default Value
	string	n/a
	Description: Identifies the Comm Plan fill from the connected radio. (One input per radio, radios 1-16)	

Control Inputs		
NetIn	Type	Default Value
	uint32	0
	Description: The current net selection of the specified radio. (One input per radio, radios 1-16)	
Receiving	Type	Default Value
	boolean	False
	Description: When true, the specified radio is actively receiving. (One input per radio, radios 1-16)	
Transmitting	Type	Default Value
	boolean	False
	Description: When true, the specified radio is actively transmitting. (One input per radio, radios 1-16)	

Audio Outputs		
AudioIn	Type	Default Value
	audio	n/a
	Description: Audio routed in from the remote client on the specified channel.	

Control Outputs		
PTT	Type	Default Value
	boolean	False
	Description: When true, audio is actively transmitting.	
InControlA	Type	Default Value
	uint8	0
	Description: Selects the radios (1-8) in the Comm Panel to transmit on. (Bitmask)	
InControlB	Type	Default Value
	uint8	0
	Description: Selects the radios (9-16) in the Comm Panel to transmit on. (Bitmask)	
NetOut	Type	Default Value
	int32	0
	Description: Net selection from the remote client. (One output per radio, radios 1-16)	
OutControlA	Type	Default Value
	uint8	0
	Description: Selects the radios (1-8) to receive from. (Bitmask)	
OutControlB	Type	Default Value
	uint8	0
	Description: Selects the radios (9-16) to receive from. (Bitmask)	

Control Outputs		
Volume	Type	Default Value
	float32	0
	Description: Sets the radio receive gain. (One output per radio, radios 1-16)	

Internal Parameters		
Default-Net	Type	Default Value
	uint8	0
	Description: The initial net that the radio will be tuned to after an install. (One parameter per radio, radios 1-16)	
Default RxState	Type	Default Value
	uint8	0
	Description: The initial receive/transmit settings after an install. (One parameter per radio, radios 1-16)	
Device Name	Type	Default Value
	string	n/a
	Description: The name given to the operator instance in the model.	
NetLock	Type	Default Value
	boolean	False
	Description: Controls whether the remote client can change the default net selection. (One parameter per radio, radios 1-16)	

Internal Parameters		
NumRadios	Type	Default Value
	uint8	0
	Description: Number of attached radios that are controlled by the remote client.	
RxLock	Type	Default Value
	boolean	False
	Description: Controls whether the remote client can change the default receive state. (One parameter per radio, radios 1-16)	

12.0. Intercoms

The Intercom components provide an audio bus service to which other components can connect to providing the ability to distribute audio throughout a model. Additionally, the intercom service and components can simulate the intercom bus structures of real aircraft and other training applications.

This section provides details on the following intercom components:

- IcomRx
- IcomTx
- Intercom_Bus_Power

This section also describes the Intercom Bus Service.

12.1. IcomRx

Summary: The Intercom Receiver component provides a connection from an intercom bus within the model. The purpose of this component is to provide a simplified means of retrieving audio from an intercom bus without the complication of a comm panel. This capability is useful for monitoring audio on a particular intercom bus within the model. As its name suggests, the Intercom Receiver component cannot transmit audio onto an intercom bus.

This component also provides the bus with a power state. If there is not an “intercom” object included for a particular bus then the bus will be fully operational by default.

Description: The Intercom Receiver component provides power conditioning of a specific bus and a direct output tap of the bus composite audio.

This component is defined to map exactly to the same connection required for a radio object. The operation of these two service-side connections are that they have values of 1.0 whenever the intercom object power is ‘True’ and 0.0 when the power is ‘False.’ Basically, the intercom bus is OFF if the power is OFF.

The intercom bus parameter defines which bus the audio is pulled from. This parameter maps to the ‘handle’ of the intercom bus.

The component output is the audio that comes from the intercom bus. The OutGain input variable controls the amplitude of the output signal. If this variable is 0.0, this component does not output a signal.

Control Input		
OutGain	Type	Default Value
	float32	1.0
	Description: OutputGain applies gain control to the signal that is retrieved from the intercom bus. Modifier: multiply Modifier_default: 1.0	

Audio Output		
OutSignal	Type	Default Value
	audio	False
	Description: Outputs audio from the selected intercom bus.	

Internal Parameter		
Channel	Type	Default Value
	id	unassigned
	Description: Channel selects the intercom bus identifier.	

12.2. IcomTx

Summary: The Intercom Transmitter component provides a connection to an intercom bus within the model. The purpose of this component is to provide a simplified means of placing audio on an intercom bus without the complication of a comm panel. This capability is useful for distributing audio from a single source that should be heard through multiple comm panels in the model. The source audio for transmission comes from an external signal connection into the Intercom Transmitter. As its name suggests, the Intercom Transmitter component cannot retrieve audio from an intercom bus.

Description: The Intercom Transmitter component provides power conditioning of a specific bus and an input for an auxiliary signal to be placed onto the bus. Only ONE intercom object may be connected to any individual intercom bus within the intercom service - any other configuration is invalid.

This component is defined to map exactly to the same connections required for a radio transmitter object.

The parameter intercom bus determines onto which bus the audio is injected. This parameter maps to the 'handle' input to the intercom service.

Since this component cannot retrieve audio from a bus, no output signal is required.

Audio Inputs		
InSignal	Type	Default Value
	audio	
	Description: Connects a signal to be routed onto the selected intercom bus for transmission.	

Control Input		
InGain	Type	Default Value
	float32	1.0
	Description: Applies gain control to the input signal before it is transmitted on the intercom bus.	

Internal Parameter		
Channel	Type	Default Value
	id	assigned
	Description: Channel selects the intercom bus identifier.	

12.3. Intercom Bus Power

Summary: Provides power for the IcomRx and IcomTx components to play audio over an intercom bus.

Description: Select the intercom bus to control power. Power toggles the Intercom Bus Power. The intercom bus is powered on by default.

Control Input		
Power	Type	Default Value
	boolean	false
	Description: Power toggles the intercom bus power. Modifier: XOR Modifier_default: True	

Internal Parameter		
Intercom-Bus	Type	Default Value
	id	unassigned
	Description: Selects the intercom bus to toggle power.	

12.4. Intercom Bus Service

Summary: The Intercom service provides an invisible audio connection between objects that declare they are attached to a common intercom ‘channel.’ All connectivity between a specific operators comm panel and the source and source/sink objects is carried out via intercom service links. The service supports multiple simultaneous channels that operate in isolation from any others. Inputs to a particular channel mix. Outputs from a channel are common to all destination objects, with one exception related to the operation of sidetone. Sidetone is the return signal from an object that has a transmit/receive capability, such as a radio. See the ‘Description’ for a full definition of the sidetone characteristic. Both half and full duplex connectivity between objects is supported depending on the source/sink object type.

Note: As with all ‘Service’ type components the user does not manually instigate the creation of the service object; this is done automatically on demand by the loader when it detects that a component has an intercom service port connection. Only one service component of the appropriate type will be loaded based on the first found need for a service of this type.

Description: The Intercom service provides an “N” channelized audio mixer and distribution service. The Intercom service will provide a centralized point for the generation of sidetone signal return to source objects. The majority of voice systems receive a portion of the source signal as a return that acts to assure the user that the system is functional. It is important however that the sidetone is generated cleanly since there are many potential issues related to multiple signal returns to an operator - when an operator should hear sidetone, the receive signal should NOT contain the same source so a return path must provide a “local loop” for the sidetone and ensure that the source signal is subtracted from the “receive” signal. This action must also accommodate any processing delays within the signal paths.

This service also includes some intelligence on allowing it to condition the return of sidetone based on state data returned from a connected object, specifically when used to link an operator to a radio or network intercom object. When configured this way the radio object must provide a sidetone ‘state’ return, that will condition whether any sidetone is to be returned to any operators that may be linked to the channel.

The Intercom service will touch many other components throughout a model, and hence it is important to define the nature of the service ‘edge’ primitive. These primitives will provide the component interface to/from the intercom service. This interface will pass the input audio to the service, and receive the return audio from the service, and will pass additional data into the service, including (but not necessarily limited to - developer to expand as required) channel index (set by the model developer Channel Handle selection), sidetone gain, audio activity status,

and perhaps component type (used to determine how the sidetone gain is used - either as a local loop gain or gate value [for radios/local or net intercom]). The ONLY user defined input to this primitive will be the selected Channel Handle. No other inputs or settings are to be required.

Available channels for use are declared through the “Intercom” tool. Within the tool a “handle” (bus name) is defined and the tool assigns an internal index ‘number’ to the channel. By default the tool assigns the channel numbers incrementally. Internally the intercom service uses the channel number to link inputs and outputs. The channel numbers are NOT to be accessible by the user, and need NOT displayed. All displays related to the handles must be in alphabetic order to aid the user locating required handles for assignment.

The Intercom tool will also support diagnostic facilities to allow the user to view the use of intercom channel within the model. This will scan the model for all objects using the service and extract the information into a table-like view. Two views will be selectable for the intercom service and are demonstrated in the following example:

Handle view – this will display all components that are connected to each handle:

UHF_Bus >

UHF_Radio > Main_Asset

OP1_Comm_Panel > Asset_Definition_1

OP2_Comm_Panel > Asset_Definition_1

VHF_Bus >

Main_Asset > UHF_Bus

OP1_Comm_Panel >

Asset_Definition_1 > UHF_Bus

Asset_Definition_2 > VHF_Bus

Asset_Definition_3 > (none)

Asset_Definition_4 > (none)

OP2_Comm_Panel

Asset_Definition_1 > UHF_Bus

Asset_Definition_2 > (none)

Asset_Definition_3 > (none)

Asset_Definition_4 > (none)

13.0. Platform

The following section details the platform components and the objects within them.

The platform components include:

- Detonation
- Fire
- Geocentric Position
- Geodetic Position
- Relative Position

13.1. Detonation

Summary: The Detonation component looks at Detonation PDUs on the DIS network and outputs their values for use within the model.

Description: This component is used primarily for triggering sounds and events based on incoming Detonation PDUs from the DIS Network. The component presents relevant fields needed to determine the sound, while other parts of the PDU are ignored. All values are held until another Detonation PDU is detected. The Serial variable will increment with every Detonation PDU detected on the network. The PDU Type must equal 3 for the Detonation component to present the values.

All nomenclature is based on the DIS standard for Detonation PDUs.

Control Outputs		
Serial	Type	Default Value
	uint32	0
	Description: The Serial value will increment with each incoming Detonation PDU, such as type 3 DIS PDUs. All other values in the component will update with the Serial value based on the incoming packet.	
Location X, Y, Z	Type	Default Value
	float32	0.0
	Description: The geocentric location, in meters.	
VelocityX, Y, Z	Type	Default Value
	float32	0.0
	Description: The geocentric velocity, in meters per second.	

13.2. Fire

Summary: The Fire component looks at Fire PDUs on the DIS Network and outputs their values for use within the model.

Description: This component is used primarily for triggering sounds and events based on incoming Fire PDUs from the DIS Network. The component presents relevant fields needed to determine the sound, while other parts of the PDU are ignored. All values are held until another Fire PDU is detected. The Serial variable will increment with every Fire PDU detected on the network. The PDU Type must be equal to 2 for the Fire component to present the values.

All nomenclature is based on the DIS standard for Fire PDUs.

Control Outputs		
Serial	Type	Default Value
	uint32	0
	Description: The Serial value will increment with each incoming Detonation PDU, such as type 2 DIS PDUs. All other values in the component will update with the Serial value, based on the incoming packet.	
Location X, Y, Z	Type	Default Value
	float32	0.0
	Description: The geocentric location, in meters.	
VelocityX, Y, Z	Type	Default Value
	float32	0.0
	Description: The geocentric velocity, in meters per second.	

13.3. Geocentric Position

Summary: The Geocentric World Position object provides a simple location feature for radio and transmitter positioning.

Description: The World positions of the transmitter and receiver are used to compute the diminishing power, as well as occulting by the earth for line of sight transmissions. The standard model of the earth is a smooth ellipsoid a terrain server maybe used for accurate modeling.

The Geocentric World position is identical to the Geodetic World Position, except that the input values for the position are given in terms of X, Y, and Z coordinates from the center of the Earth in meters.

If the world position is 0, 0, 0 (i.e. the center of the Earth), then the ranging effects of any attached radio are turned off and the radio will clearly receive all transmissions on its frequency.

Control Inputs		
X	Type	Default Value
	float64	0.0
	Description: The X world position coordinate, units are in meters.	
Y	Type	Default Value
	float64	0.0
	Description: The Y world position coordinate, units are in meters.	
Z	Type	Default Value
	float64	0.0
	Description: The Z world position coordinate, units are in meters.	

Internal Parameter		
World Position Bus	Type	Default Value
	id	unassigned
	Description: Selects the world position bus to assign a position.	

13.4. Geodetic Position

Summary: The Geodetic World position component provides a simple location feature for the radio and transmitter positioning.

Description: The World positions of the transmitter and receiver are used to compute diminishing power and occulting by the earth for the line of sight transmissions. The model of the earth is a smooth ellipsoid (model WGS84).

The Geodetic position is specified in altitude (in meters) with latitude, and longitude in degrees.

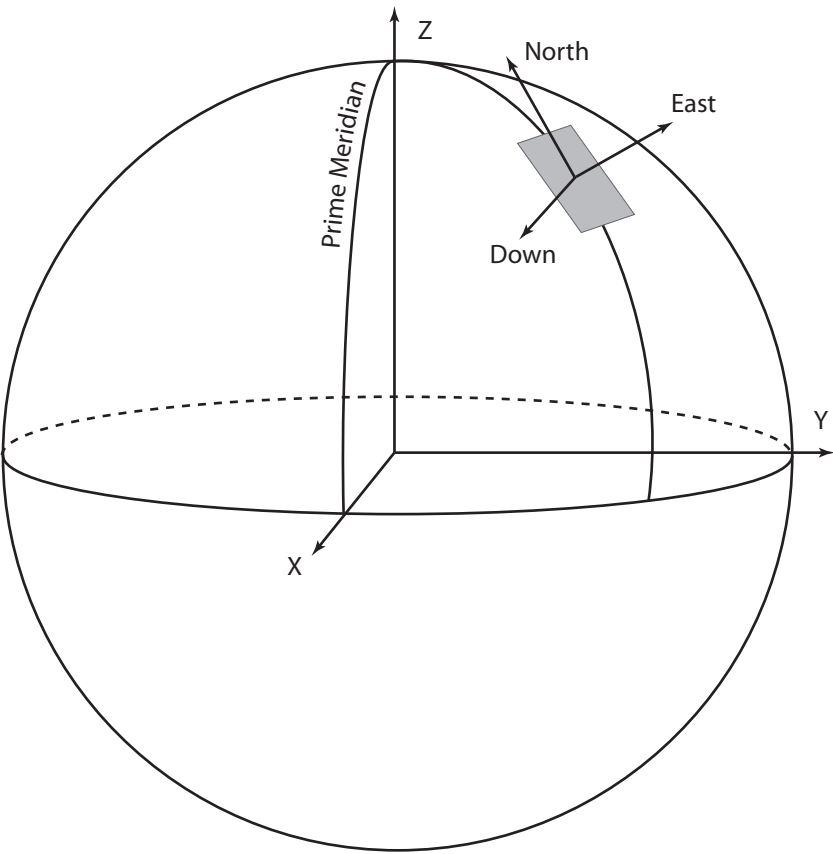
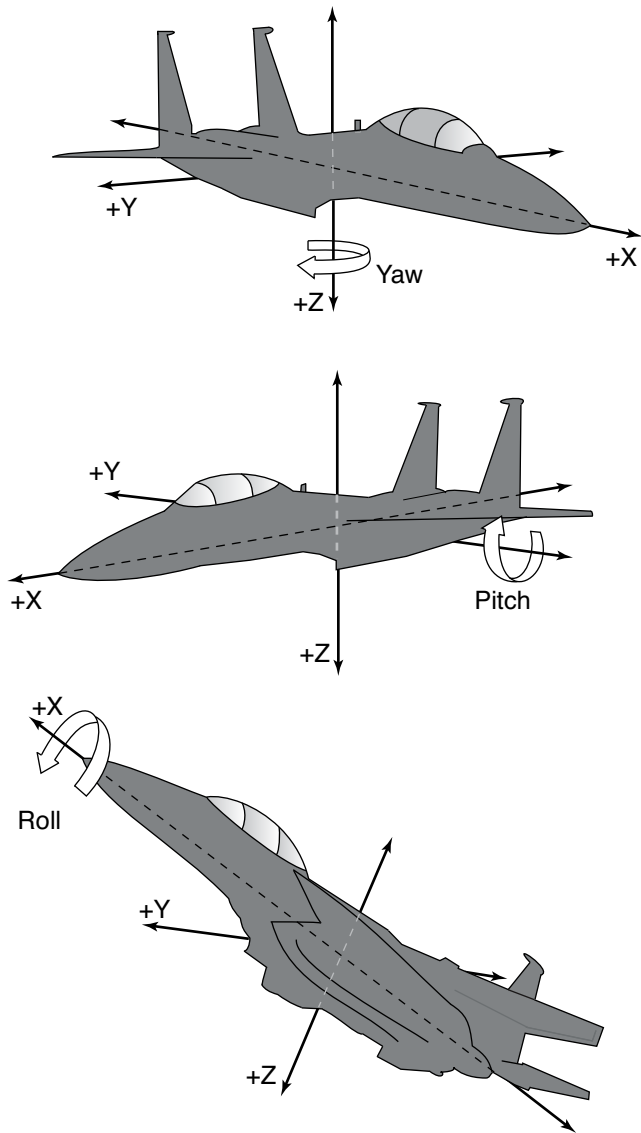
Control Inputs		
Elevation	Type	Default Value
	float32	0.0
	Description: The altitude world position coordinate, units are in meters.	
Latitude	Type	Default Value
	float32	0.0
	Description: The latitude world position coordinate, units are in degrees.	
Longitude	Type	Default Value
	float32	0.0
	Description: The longitude world position coordinate, units are in degrees.	

Internal Parameters		
World Position Bus	Type	Default Value
	id	unassigned
	Description: Selects the world position bus to assign a position.	
Position	Type	Default Value
	woldposn_geocentric	0.0
	Description: Selects the world position X,Y, Z coordinates.	

13.5. Relative Position

Summary: The Relative Position component calculates the relative coordinates and velocities of an entity, used with the Highway 3D Service and other orientation-dependent sound models.

Description:



Control Inputs		
Ent_Alt	Type	Default Value
	float64	1.0
	Description: The altitude of the entity position.	

Control Inputs		
Ent_Lat	Type	Default Value
	float64	1.0
	Description: The latitude of the entity position.	
Ent_Lon	Type	Default Value
	float64	1.0
	Description: The longitude of the entity position.	
Ref_Alt	Type	Default Value
	float64	1.0
	Description: The altitude of the reference position.	
Ref_Lat	Type	Default Value
	float64	1.0
	Description: The latitude of the reference position.	
Ref_Lon	Type	Default Value
	float64	1.0
	Description: The longitude of the reference position.	
Ref_Yaw	Type	Default Value
	float64	1.0
	Description: The measure of the angle formed from True North to the entity's +X axis. This angle is specified in degrees and is positive clockwise if looking along the +Z axis.	

Control Inputs		
Ref_Pitch	Type	Default Value
	float64	1.0
	Description: The measure of the angle between the reference plane and the entity's +X axis. This angle is specified in degrees and is positive above (away from the ellipsoid) the reference plane.	
Ref_Roll	Type	Default Value
	float64	1.0
	Description: The measure of the angle between the reference plane and the entity's +Y axis along a plane perpendicular to the entity's X axis. It is the angle of rotation about the X axis after the yaw and pitch have been applied. Roll is specified in degrees and is positive clockwise from the point of view looking along the +X axis.	

Control Outputs		
Approach Speed	Type	Default Value
	float32	0.0
	Description: The rate at which the entity and reference positions are approaching each other in meters per second.	
Distance	Type	Default Value
	float64	0.0
	Description: Distance from the reference point to the entity in meters.	

Control Outputs		
Ent_X	Type	Default Value
	float64	6378137
	Description: The X position of the entity in the geocentric coordinate system.	
Ent_Y	Type	Default Value
	float64	0.0
	Description: The Y position of the entity in the geocentric coordinate system.	
Ent_Z	Type	Default Value
	float64	0.0
	Description: The Z position of the entity in the geocentric coordinate system.	
Ent_Speed	Type	Default Value
	float32	0.0
	Description: The instantaneous speed of the entity in meters per second.	
Ref_X	Type	Default Value
	float64	6378137
	Description: The X position of the reference point in the Geocentric coordinate system. Units are in meters.	

Control Outputs		
Ref_Y	Type	Default Value
	float64	0.0
	Description: The Y position of the reference point in the Geocentric coordinate system. Units are in meters.	
Ref_Z	Type	Default Value
	float64	0.0
	Description: The Z position of the reference point in the Geocentric coordinate system. Units are in meters.	
Rel_X_Pos	Type	Default Value
	float64	0.0
	Description: The X position of the entity relative to the position and orientation of the reference frame.	
Rel_Y_Pos	Type	Default Value
	float64	0.0
	Description: The Y position of the entity relative to the position and orientation of the reference frame.	
Rel_Z_Pos	Type	Default Value
	float64	0.0
	Description: The Z position of the entity relative to the position and orientation of the reference frame.	

Control Outputs		
Rel_X_Vel	Type	Default Value
	float32	0.0
	Description: The relative velocity in the X direction between the reference and entity positions.	
Rel_Y_Vel	Type	Default Value
	float32	0.0
	Description: The relative velocity in the X direction between the reference and entity positions.	
Rel_Z_Vel	Type	Default Value
	float32	0.0
	Description: The relative velocity in the X direction between the reference and entity positions.	

Debugging Variables			
E_Axis	Variables	Type	Default Value
	X	float64	0.0
	Y	float64	1.0
	Z	float64	0.0
	Description: The vector pointing East in the local geographic frame of the reference position.		

Debugging Variables			
D_Axis	Variables	Type	Default Value
	X	float64	-1.0
	Y	float64	0.0
	Z	float64	0.0
	Description: The vector pointing Down in the local geographic frame of the reference position.		
N_Axis	Variables	Type	Default Value
	X	float64	0.0
	Y	float64	0.0
	Z	float64	1.0
	Description: The vector pointing North in the local geographic frame of the reference position.		
Vec_to_Entity	Variables	Type	Default Value
	X	float64	0.0
	Y	float64	0.0
	Z	float64	0.0
	Description: Shows the direction to the entity from the reference position.		
X_Axis	Variables	Type	Default Value
	X	float64	0.0
	Y	float64	0.0
	Z	float64	1.0
	Description: The vector pointing Forward from the reference position. For example, this is the vector pointing through the nose of the airplane, if the airplane is the reference.		

Debugging Variables			
Y_Axis	Variables	Type	Default Value
	X	float64	0.0
	Y	float64	1.0
	Z	float64	0.0
	Description: The vector pointing to the right from the reference position.		
Z_Axis	Variables	Type	Default Value
	X	float64	-1.0
	Y	float64	0.0
	Z	float64	0.0
	Description: The vector pointing down from the reference position.		

14.0. Host Control

Before adding the HostIn and HostOut components, you must first add a host model. This is required in order to create the host I/O packets. Inside the host model, select to add either a HostIn or HostOut controller and then add the host I/O components also known as packets.

The most common question asked regarding host control is how to debug it and find out if it is working properly. In order to debug the host control, the user must figure out if packets are coming in off the network from the host.

To determine packet activity, there are two items to pay close attention to in the host interface. The first is 'Live Capture' which takes a screen capture of the packet activity coming in from the host and the second is the 'Controller' which displays the packet statistics.

In the 'Controller,' also called the ACE HostIn Viewer, the Fail Count column will always show data if the host is working, the numbers will continue to increment in a loop if it is working properly. The 'Total Packets Received' number will increment over time if the host control is working properly.

The TestIn and TestOut components are for ASTi internal testing, contact ASTi for more information.

14.1. HostIn

Summary: The Host Input component serves as a user interface wrapper for control data fields extracted from external sources (including host Ethernet control UDP packets and state machine cells), providing a means to apply incoming control data to other components in the model. This describes the Host Input as used with the host Ethernet. A future revision of the Host Input component may include an interface to state machine cells.

Users create and modify Host Input components using the ACE IO Packet Editor. With this tool users specify the source of the data packet by UDP port, then extracted individual Host Input components are connected to other model components, serving as control variables.

Description: The IO Packet Editor provides a means to functionally disassemble a host UDP packet into individual fields. The Host Input component is the composite collection of individual data fields in a specific packet. The individual fields within the Host Input (data sources) are used as control variables by other model components (data sinks).

The Host Input component is analogous to the various Control objects in Model Builder. The fundamental difference between the Host Input component and (MB) control object is: whereas MB included a separate control object for each data type, the Host Input component encompasses all fields within a host packet. Additionally, the Host Input accommodates all available data types.

The process for creating and using Host Inputs follows:

1. Using the IO Packet Editor, specify the UDP port of the host packet and the most-least significant data order (big or little endian).
2. To create a field within a Host Input component, specify the parameters defining the data field: offset byte location within the packet (or in the case of a boolean field, offset byte and bit), the data type and the initial value for the field.

Once a Host Input field (data source) has been created, it can be linked to a variable (data sink) inside another component. The action of making a link is done on the sink side of the link. The link is visible from the source side (the Host Input "Used By" shows the sink component name) and from the sink side (the sink component "From" shows the Host Input field name).

Each field in the Host Component is defined by:

A. Name – The Name field is used as a descriptor for the component.

B. Offset – The location of the data field within the packet, defined by the byte offset from the start of the data field. The location of a boolean input is further defined by the bit position within a particular byte.

C. Message Length – Packet length in bytes.

D. Type – The Host Input can be set to one of these data types:

Unsigned Integer, 8-Bit (uint8)CString

Unsigned Integer, 16-Bit (uint16)ident

Unsigned Integer, 32-Bit (uint32) bits8_0 to bits8_7

Unsigned Integer, 64-Bit (uint64)message

Signed Integer, 8-Bit (int8)

Signed Integer, 16-Bit (int16)

Signed Integer, 32-Bit (int 32)

Signed Integer, 64-Bit (int64)

Floating Point, 32-Bit (float32)

Floating Point, 64- Bit (float64)

Boolean, 1-Bit

E. Initial Value – The component's initial value is set by the application should the source packet not be present (due to host failure). This is directly analogous the initial value featured in MB Control objects.

F. Function – Add a basic function including Add, Linear, Logical-and, Logical-or, Logical-xor, Multiple, and Subtract.

G. Y – Value added if a function is used.

H. Z – Value added if a function is used.

I. Rscale – Adds a scale factor to apply to the gain output of the function.

J. Test Mode – A local test mode is provided to override the incoming value from the host packet and manually set the value of a field. Test modes for various fields can be selectively enabled. A master test mode enable switch activates the test mode for all test mode-enabled fields. All fields which are not test mode-enabled continue to get their value from either the host packet or their initial value (if the host has failed). The application checks test mode values for correct range. If an out-of-range test mode value is entered, the application will return a default value.

Test Mode Selections: ON or OFF

Integers, all types:

Test ModeBoolean, Default Value = OFF

ValueInteger, Default Value = 0

Ramp Rate (Hz)Floating Pt, Default Value = 0.0

Min Ramp ValueInteger, Default Value = 0

Max Ramp Value Integer, Default Value =0

Floats, all types:

Test ModeBoolean, Default Value = OFF

ValueFloating Pt, Default Value = 0

Ramp Rate (Hz)Floating Pt, Default Value = 0.0

Min Ramp ValueFloating Pt, Default Value = 0.0

Max Ramp ValueFloating Pt, Default Value = 0.0

Boolean:

Test Mode Boolean, Default Value = OFF

Value Boolean, Default Value = 0

Toggle Rate (Hz)Floating Pt, Default Value = 0.0

Mark/Space Floating Pt, Default Value = 0.5

For integers and floats:

If Test Mode = ON and Ramp Rate = 0, value sets a static test value. If test Mode = ON and Ramp Rate >0, value is overridden and a dynamic test value periodically ramps from the Min Ramp value to the Max Ramp Value. The Ramp Rate is in Hertz.

For Booleans:

If Test Mode = ON and Toggle Rate = 0, value sets a static test value. If Test Mode = ON and Toggle Rate >0, value is overridden and a dynamic test value toggles between 1 and 0. The Toggle Rate is in Hertz. The Mark/Space ratio controls the duration the boolean is 1 relative to the period of the entire 1/0 toggle cycle. (E.g. a value of 0.5 means in a given period, the boolean output will be 1 for half of the time and 0 for the other half).

K. Test Value – Displays the realtime value of the Host Input field, based on parameters set by the user (offset byte, data type, etc.). This value can be from: the host packet, initial value or the test mode value.

1. The Host Input component is connected to two different Audio Waves.
2. The first field in the UDP packet is sent to the frequency variable of the Audio Wave with the value of 100.00 as float32 and the second field in the UDP packet is sent to the gain variable in a different Audio Wave with the value 3.0 as float32. Currently, the Init. Value is set in these components because there are no UDP packets from the outside world.
3. The Audio Wave data viewers show the selected row of the variable that is getting its value from the Host Input component.

L. Used By – The Used By field defines the connection to a control sink point (Variable) in another component. Note that the value of the Host Input can be modified (inverted, scaled, and offset) at the link sink point.

M. Other – Used to edit the test parameters specifically the ramp settings including mode, frequency, minimum, and maximum.

N. Description – Comments field.

Control Outputs		
N/A	Type	Default Value
	Control Data	n/a
	Description: The Host Input output value is based on parameters set by the user (offset byte, data type, etc.).	

Internal Parameters		
BigEndian	Type	Default Value
	boolean	false
	Description: Changes the byte order. The endianness defines the byte order for the data in a packet.	
Default_on_init	Type	Default Value
	float	0.000
	integer	0
Default_on_src_fail	boolean	0
	Description: Realtime value of data field, from host packet, initial value or test mode	
	Modifier: None	
Enable_Test_Mode	Range: Min - Max values of each type	
	Type	Default Value
	float	0.000
Enable_Test_Mode	integer	0
	boolean	0
	Description: Initial value of data field, applied if source data is not present (host fail), field-wide	
Enable_Test_Mode	Modifier: None	
	Range: Min - Max values of each type	
	Type	Default Value
Enable_Test_Mode	boolean	true
	Description: True toggles the test mode on for host control testing.	

Internal Parameters		
FailTime-out	Type	Default Value
	uint8	1
	Description: Sets the threshold for assumed communication failure when a packet has not arrived within the set amount of time.	
Rxpacket-Count	Type	Default Value
	uint32	0
	Description: Displays the received packet amount.	
Source-Failed	Type	Default Value
	boolean	True
	Description: True when no activity is received from the host.	
Source-FailCount	Type	Default Value
	uint32	1
	Description: The count when no activity is received from the host.	

14.2. Host Out

Summary: The Host Out component provides a way to send control values from the model to an destination host computer via the system's ethernet interface. A future revision of the Host Out component will include transmission of model control to embedded state machines via cells.

Users create and modify Host Out components using the ACE IO Packet Editor. With this tool users specify the outgoing host data packet by UDP port, then link model control sources to individual control fields within the outgoing packet. Model control sources include any component with a control output, like: host input components, math components and component status fields.

Description: The IO Packet Editor provides a means to functionally assemble a host UDP packet from individual model controls. The Host Out component is a composite collection of specific data fields from model controls, packed into an outgoing ethernet packet.

The Host Out component is analogous to the various control objects in Model Builder. The fundamental difference between the Host Out component and (MB) control objects is:

1. Whereas MB included a separate control object for each data type, the Host Output component encompasses all fields within an outgoing packet.
2. Additionally the Host Out accommodates all available data types.

The process for creating and using Host Out follows:

1. Using the IO Packet Editor, specify the UDP port of the destination host computer and the most-least significant data order (big or little endian).
2. To create a field within a Host Out component, specify the parameters defining the data field (offset byte location within the packet, or in the case of a boolean field, offset byte and bit, and the data type) and link to a model control source.

Each field in the Host Out Component is defined by:

A. Name – The name field is used as a descriptor for the component.

B. Offset – The location of data field within packet, defined by the byte offset from the start of the data field. The location of a boolean input is further defined by the bit position within a particular byte.

C. MsgLen – Packet length in bytes.

D. Type – The Host Output can be set to one of these data types:

Unsigned Integer, 8-Bit (uint8)

Unsigned Integer, 16-Bit (uint16)

Unsigned Integer, 32-Bit (uint32)

Unsigned Integer, 64-Bit (uint64)

Signed Integer, 8-Bit (int8)

Signed Integer, 16-Bit (int16)

Signed Integer, 32-Bit (int 32)

Signed Integer, 64-Bit (int64)

Floating Point, 32-Bit (float32)

Floating Point, 64- Bit (float64)

Boolean, 1-Bit

E. Used by – The source field defines the link to a model control source.

F. Description – Comments field.

Control Outputs		
N/A	Type	Default Value
	Control Data	n/a
	Description: The Host Output output value is based on parameters set by the user (offset byte, data type, source, etc.).	

Internal Parameters		
BigEndian	Type	Default Value
	boolean	false
	Description: Changes the byte order. The endianness defines the byte order for the data in a packet.	

14.3. Cell Service

Summary: The Cell service provides an invisible data connection between objects that declare they are attached to a common cell ‘channel.’ All connectivity between a specific cell channel and the source and source/sink objects is carried out via cell service links.

Note: As with all ‘Service’ type components the user does not manually instigate the creation of the service object; this is done automatically on demand by the loader when it detects that a component has a cell service port connection. Only one service component of the appropriate type will be loaded based on the first found need for a service of this type.

Description: The ONLY user defined input to this primitive will be the selected Bus Handle. No other inputs or settings are required. Available buses for use are declared through the “CellService” tool. Within the tool a “name” (bus name) is defined and the tool assigns an internal index ‘number’ as a bus ID. By default the tool assigns the bus ID numbers incrementally. Internally the cell service uses the bus ID number to link inputs and outputs.

14.3.1. CellIn

Summary: The Cell Input component serves as a user interface wrapper for control data fields extracted from external sources (including host Ethernet control UDP packets and state machine cells), providing a means to apply incoming control data to other components in the model.

Description: The Cell Input component is a collection of 40 individual data bytes in a specific packet. The individual bytes within the Cell Input (data sources) are used as control variables by other model components (data sinks).

Control Inputs		
CellData	Type	Default Value
	uint8	0
	Description: The CellData field can only be a collection of 40 unsigned bytes, each of which can be used as a control to connect to other components within the model.	
NewCell-Data	Type	Default Value
	boolean	False
	Description: When True new cell data UDP packets are recognized. Note: This input was created for specific use by the LS653 state machine.	

Internal Parameters		
CellBusID	Type	Default Value
	id	n/a
	Description: Assigns to Cell Bus Service linking components together.	
CellRe- ceiveCount	Type	Default Value
	uint32	0
	Description: The count of the number of cells received from the cell daemon.	

14.3.2. CellOut

Summary: The Cell Out component transmits bytes of data generated by specific components.

Description: Cell Out provides a way to send control values from the model to a destination host computer via the system, using an ethernet interface. The Cell Out component is a composite collection of specific data bytes from model controls, packed into an outgoing ethernet packet.

Control Output		
CellData	Type	Default Value
	uint8	0
	Description: The CellData field can only be a collection of 40 unsigned bytes, each of which can be used as a control to connect to other components within the model.	

Internal Parameters		
CellBusID	Type	Default Value
	id	n/a
	Description: Assigns to Cell Bus Service linking components together.	
CellTransmitCount	Type	Default Value
	uint32	0
	Description: The count of the number of cells transmitted from the cell daemon.	

15.0. Radio Components

Building radios in ACE is slightly different from the traditional modeling used in other ASTi generations of software. A basic radio is now made up of two components, a Transceiver and a Radio Control Unit (RCU). The two objects are analogous to live radios on many aircrafts. Typically, there is a base unit which is responsible for the over-the-air transmission known as the RT and control heads that serve as the interface for the pilot or co-pilot.

The idea in ACE is that every radio requires use of the Transceiver component, a master object which can be used for any radio, but it contains a minimum of settings. This way, if the user wants to simulate two different types of radios, the RCU is used to customize the Transceiver to be an ARC-210 versus an ARC-232.

The third piece of the radio is the fill that it receives from the CommPlan located in the Layout of the project. The CommPlan provides a flexible way of adjusting internal radio parameters such as Modulation Type or the Transmit Gain. Previous generations of ASTi software accomplished the fill as a Mode Table within the Radio component. With the fills placed in the CommPlan, it is now easy to share mode settings across several radios inside a model. In addition to the different mode settings, the fill of a radio can also set default frequencies.

The most commonly configured radio uses two components, the Transceiver and RCU_Basic. The RCU provides a fill from the CommPlan as well as a suite of parameters which can override their counterparts from within the fill. For example, RCU_Basic has a loaded fill and is using Net 1, which sets the frequency of the radio to 101MHz. This information is sent down to the Transceiver via the Transceiver ID located in both components. But suppose a user wanted to have every setting inside the net remain the same, except that they wanted to use a new frequency of 105MHz. By setting the Frequency of the RCU to 105Mhz, that will override the fill, and the Transceiver will act accordingly.

A radio can switch between any number of nets, defined in the communications plan, each of which can be custom tailored to provide control over parameters such as modulation, noise, bandwidth, crypto system and key, frequency hopping net ID, SATCOM, and other parameters. The default communications plan fill will support the most common modes used by real RF radios, including UHF, VHF, HF, SINCGARS and HaveQuick. This allows the user to get started quickly, while retaining the flexibility to further fine tune the simulation. The Transceiver is also capable of receiving and transmitting TDL messages. Various voice-encoding schemes are also supported including CVSD, mu-law and PCM.

In general, settings that relate to how the radio appears on the network are in the Transceiver, and settings which customize the radio for simulation are placed in the RCU. For more information about building a basic radio, see the radio tutorials inside the T4 Training Manual (DOC-01-TEL4-TM-1).

The following section details the radio components and the objects within them.

The radio components include:

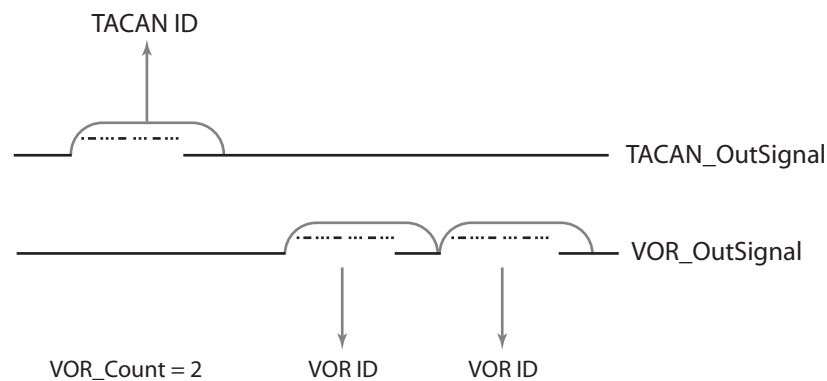
- ColocatedBeacon
- GenericControl
- HFServer
- Intercom Transceiver
- ICU
- MarkerTone
- MorseKeyer
- RCUbasic
- RCUCryptokey
- RCUfrequency
- RCUhavequick
- RCUoverride
- RCUincgars
- RCUtxpower
- Receiver
- Relay
- Satellite
- Transceiver
- Transmitter
- VORTAC_Controller

15.1. ColocatedBeacon

Summary: The ColocatedBeacon component creates the audio to simulate Co-located VOR and TACAN transmitters. Morse Identifier tones are generated for both transmitters.

Description: Co-located VOR and TACAN transmitters present a special case when simulating navigational aides. Both transmitters will produce a Morse Identifier that will be constantly broadcast over the air, but the IDs are broadcast in sequence. As a result, the ColocatedBeacon component can be thought of as two separate Morse Keyers which are designed to never play at the same time.

The basic timing can be seen in the picture below.



Control Inputs		
Enable	Type	Default Value
	boolean	True
Description: When True ColocatedBeacon is enabled.		

Control Inputs		
Ident	Type	Default Value
	ident	n/a
	Description: The ASCII characters that drive the MorseKeyer. Ident is a special variable Type which is defined as 4 ASCII characters concatenated together. A HostIn component must be used to drive this parameter.	
Interval	Type	Default Value
	float32	1
	Description: Used to set the delay between words. Units are in seconds.	
TACAN_Frequency	Type	Default Value
	float32	1.0
	Description: Provides the frequency (pitch) of the TACAN Morse tone. Units are in Hertz.	
TACAN_Gain	Type	Default Value
	float32	1.0
	Description: Scales the strength of the generated tone of the TACAN output signal.	

Control Inputs		
TACAN_I LSPause	Type	Default Value
	boolean	False
	Description: ILS Pause places a space between the first and second characters of the Ident. When FALSE, the TACAN Keyer uses normal morse code inter-character timing which is a 3 dot gap between all Ident characters. When TRUE, the TACAN Keyer uses a standard morse code inter-word timing between the first and second characters (i.e. a 7 dot gap). Inter-character timing for subsequent Ident characters will use standard morse code inter-character spacing (i.e. a 3 dot gap).	
VOR_ Count	Type	Default Value
	uint16	1
	Description: Number of times the VOR Identifier is played before keying the TACAN identifier. The VOR and TACAN identifier tones are mutually exclusive.	
VOR_ Frequency	Type	Default Value
	float32	1.0
	Description: Provides the frequency (pitch) of the VOR Morse tone. Units are in Hertz.	
VOR_ Gain	Type	Default Value
	float32	1.0
	Description: Scales the strength of the generated tone of the VOR output signal.	

Control Inputs		
VOR_ILS Pause	Type	Default Value
	boolean	False
	Description: ILS Pause places a space between the first and second characters of the Ident. When FALSE, the VOR Keyer uses normal morse code inter-character timing which is a 3 dot gap between all ident characters. When TRUE, the VOR Keyer uses a standard morse code inter-word timing between the first and second characters (i.e. a 7 dot gap). Inter-character timing for subsequent ident characters will use standard morse code inter-character spacing (i.e. a 3 dot gap).	
Wordrate	Type	Default Value
	uint8	1
	Description: Determines the rate at which the word is keyed. Units are in dots per second. The faster the rate the higher the number.	

Audio Outputs		
TACAN_ OutSignal	Type	Default Value
	audio/audio	n/a
	Description: Outputs the keyed word for TACAN radio at appropriate intervals when the ColocatedBeacon is enabled.	
VOR_Out Signal	Type	Default Value
	audio/audio	n/a
	Description: Outputs the keyed word for VOR radio at appropriate intervals when the ColocatedBeacon is enabled.	

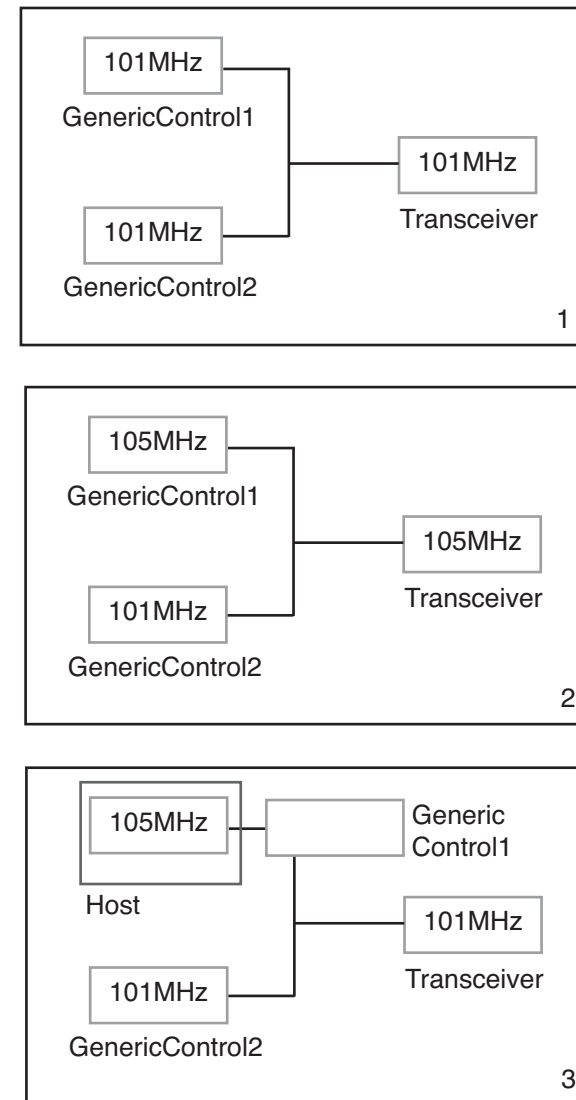
Control Outputs		
TACAN_ Active	Type	Default Value
	boolean	False
	Description: When keyer is keying a dot or a dash this goes True, essentially mimicking the ‘beep’ of the keyer.	
VOR_ Active	Type	Default Value
	boolean	False
	Description: When keyer is keying a dot or a dash this goes True, essentially mimicking the ‘beep’ of the keyer.	
TACAN_ Busy	Type	Default Value
	boolean	n/a
	Description: This flag indicates if the TACAN keyer is actively producing Morse audio.	
VOR_ Busy	Type	Default Value
	boolean	n/a
	Description: This flag indicates if the VOR keyer is actively producing Morse audio.	

15.2. Generic Control

Summary: Generic radio control for the radio simulated environment.

Description: The Generic Control component is an RCU that is designed for use in the case where multiple RCUs are hooked onto the same Transceiver. It mimics the RCU_Basic component with one key exception, any updates to the parameters of the RCU are sent to the Transceiver on change, instead of all the time. This allows the model to have two Generic Control components, each with a different frequency, correctly driving the Transceiver.

At any given time, the Transceiver will take the last sent value for a parameter and ignore those same settings on other RCUs.



The diagram above illustrates Generic Controls changing the frequency of a transceiver in time.

Control Inputs		
Crypto Key	Type	Default Value
	uint16	0
	Description: If two radios are using encryption then they must have matching crypto keys for the crypto modes. This field must match in order for proper radio tuning.	
Fill	Type	Default Value
	fill	n/a
	Description: Insert the fill created in the CommPlan by double-clicking under the 'Value' column.	
FreqHop NetID	Type	Default Value
	uint16	0
	Description: Identifies the frequency hopping network.	
FreqHop SyncTOD	Type	Default Value
	uint32	0
	Description: Identifies the Time of Day used in frequency hopping.	
FreqHop Transec Key	Type	Default Value
	uint16	0
	Description: Identifies the transmission security key used in generating hopping patterns.	
FreqHop HopSet-WOD	Type	Default Value
	uint16	0
	Description: Identifies the set of frequencies used in hopping pattern.	

Control Inputs		
FreqHop LockOutId	Type	Default Value
	uint16	0
	Description: Identifies the set of frequencies that are excluded from hopping pattern.	
FreqHop System	Type	Default Value
	uint16	0
	Description: Sets the system type to enable frequency hopping such as HaveQuick or SINCGARS. When 0 frequency hopping is ignored.	
Frequency	Type	Default Value
	uint64	0
	Description: The current radio tune frequency in Hz.	
Net	Type	Default Value
	uint32	1
	Description: Net defines the core radio features including frequency, Tx frequency, waveform, crypto, and frequency hopping. The Net parameters are set in the Comm Plan.	
TransceiverID	Type	Default Value
	id	unassigned
	Description: TransceiverID tells the RCU what Transceiver component it should control.	

Control Inputs		
Tx Frequency	Type	Default Value
	uint64	0
	Description: The transmitter frequency of the incoming signal. All frequencies are in Hz. For use only if the transmit frequency differs from the receive frequency.	
TxPower	Type	Default Value
	float32	0.0
	Description: The transmit power for the radio.	

15.3. HFServer

Summary: The HF Server component allows control of the HF Server Application providing real-time, high-fidelity modeling of HF radios.

Description: The HF Server computes propagation effects between virtual radios, taking into account such things as transmitter-receiver global position, season, time of day (day-night terminator), and solar activity.

To properly simulate solar activity and seasonal/circadian effects on the ionosphere and HF radio signal propagation, the HF Server requires that the Smoothed Sunspot Number (SSN) and Time-of-Day offset be set. These parameters are set through use of the HF Server component

Note: Configuration parameters related to the HF Server's network behavior including the interface and UDP port number to send/receive HF requests are set via the DIS Gateway configuration in the Project.

Control Inputs		
SSN	Type	Default Value
	uint32	0
	Description: Smoothed Sunspot Number. Typical values of the SSN range from 0 to 250, depending on past and current sunspot activity. The default value for this variable is 100.	

Control Inputs		
Offset	Type	Default Value
	int32	0
	Description: Time-of-Day offset in seconds between the HF Server clock (local time in GMT) and the simulation time (in GMT). For example, if the exercise is being conducted on the East coast of the U.S., during summer (GMT +5), at the location's 8:00am (08:00 Eastern Standard Time), the HF Server's local clock would read 13:00 GMT. If the <i>simulated scenario</i> is taking place on the West coast of the U.S., during summer (GMT +8) at the location's 3:30pm (15:30 Pacific Standard Time), then a clock in the simulated world would read 23:30 GMT. The difference between the simulated GMT (23:30) and the real-world GMT (13:00) is the Time-Of-Day offset; namely, +10.5 hours. Using this control, a user can force the HF Server to return results for a night-time mission, even though the exercise is taking place during the day. Likewise, seasonal effects typical of winter propagation can be used in July. The default value for the time offset is zero hours.	
Domain-NameIn	Type	Default Value
	string	n/a
	Description: Allows a host platform to configure the Domain name string remotely. This input is typically used in conjunction with the Control> NumtoString component. Refer to the DomainName string definition for syntax.	

Internal Parameters		
Domain-Name	Type	Default Value
	string	n/a
Description: The DomainName is part of the ASTiNet property set and defines a common communications environment. All radios within the same domain have the ability to communicate and those radios in separate domains can never communicate. It is analogous to a DIS Exercise ID or an HLA Federation name. When defined as an ASCII string the Transceiver is natively an ASTiNet radio. When that string matches a defined DIS domain (at the project level) it will map the domain name to a DIS exercise ID. As a shortcut, define a DIS Exercise ID by entering “DIS:N”, where N=1 through 255. For example, “DIS:1” would but the radio in DIS Exercise ID #1. Important: This parameter is required. In the HF Server component, this specifies the for which the HF Server configuration is valid. Multiple Domains can be controlled by using multiple HF Server components.		

15.4. Intercom Transceiver

Summary: The Intercom Transceiver is a simple version of the Transceiver component and is used with the ICU component to create a network intercom.

Description: The Intercom Transceiver connects to an ICU to create a network intercom that broadcasts audio over the DIS network. Unlike the Transceiver, the Intercom Transceiver operates as an Intercom only. It does not have the full radio features.

Control Inputs		
Domain-NameIn	Type	Default Value
	string	n/a
	Description: Allows a host platform to configure the Domain name string remotely. This input is typically used in conjunction with the Control > NumtoString component. Refer to the DomainName string definition for syntax.	
PowerIn	Type	Default Value
	boolean	True
	Description: Enable or disable power for the Intercom Transceiver. If not connected in the “From” field, default value is true.	
ProtocolIDIn	Type	Default Value
	string	n/a
	Description: Allows a host platform to configure the ProtocolID string remotely. This input is typically used in conjunction with the Control > NumtoString component. Refer to the ProtocolID definition for syntax.	

Control Outputs		
Local Audio	Type	Default Value
	boolean	false
	Description: Connects audio for transmit or receive. Also connects the Tune tone which plays first before the remaining audio.	
RxActive	Type	Default Value
	boolean	false
	Description: Indicates whether the Intercom Transceiver is receiving.	
TxActive	Type	Default Value
	boolean	false
	Description: Indicates whether the IntercomTransceiver is transmitting. When True the IntercomTransceiver is transmitting.	

Internal Parameters		
Audio	Type	Default Value
	uint32	0
	Description: Displays the encoding rate, encoding type and the packet samples.	

Internal Parameters		
Domain-Name	Type	Default Value
	string	
	Description: The DomainName is part of the ASTiNet property set and defines a common communications environment. All transceivers within the same domain have the ability to communicate and Transceivers in separate domains can never communicate. It is analogous to a DIS Exercise ID or an HLA Federation name. When defined as an ASCII string the Transceiver is natively and ASTiNet Radio. When that string matches a defined DIS domain (as the project level) it will map the domain name to a DIS exercise ID. As a shortcut, define a DIS Exercise ID by entering “DIS:N”, where N = 1 through 255. For example, “DIS:1” would put the Transceiver in DIS Exercise ID #1.	
Intercom-Bus	Type	Default Value
	id	unassigned
	Description: Assigns the Intercom Transceiver to the Intercom Bus Service. The Intercom Bus Service allows for input audio, sidetone audio, and output audio to be passed around among components that are connected to the bus.	
Intercom-Name	Type	Default Value
	string	
	Description: Set an identification name for the Intercom Transceiver.	
Mode	Type	Default Value
	TunerMode	none
	Description: Displays whether the Intercom Transceiver is in VoIP or Intercom mode. Radio modes not supported.	

Internal Parameters		
NetName	Type	Default Value
	string	
	Description: Indicates the set Net name.	
PowerBus	Type	Default Value
	id	unassigned
	Description: Connects to the Power Service and is used in conjunction with Power Service to receive power instead of using a control.	
Power-State	Type	Default Value
	boolean	true
	Description: Displays the power state of the Intercom Transceiver.	

Internal Parameters		
ProtocolID	Type	Default Value
	string	
	Description: The Protocol ID field has a variety of use cases. It allows a radio to become a DIS radio by setting the DIS identifiers such as the Host ID and Radio ID. The ProtocolID also allows the Marking Field to be set, or for the radio to be considered "local" and not published on the network. DIS:### - Sets the Site, Application, Entity, and Radio IDs DIS:## - Sets the Entity and Radio ID DIS:# - Sets the Radio ID DIS: - Sets all four IDs to be set automatically LCL: - Sets the radio to be Local and not published on the network. If any of the above numbers are not set, the remaining Identifiers are generated automatically. The Site and Application ID are set via the last two octets of the DIS IP Address. The Entity and Radio IDs are generated randomly. The above examples can be appended with another colon and the DIS Marking Field. For example, DIS:100.3:RedForce1, sets the Entity ID to 100, the Radio ID to 3, and the Marking Field to RedForce1. The Site and Application IDs will be set from the IP address.	
RxChannel	Type	Default Value
	uint64	0
	Description: Reflects the receive channel set in the ICU.	

Internal Parameters		
RxAudio	Type	Default Value
	boolean	False
	Description: The IntercomTransceiver local audio receive audio stream, i.e. this is the receiving audio the Intercom Transceiver will pickup from transmitting intercoms.	
Sidetone	Type	Default Value
	boolean	False
	Description: Sidetone audio signal generated by mixing all the received sidetones from the actively selected Intercom busses.	
TransceiverID	Type	Default Value
	id	unassigned
	Description: Defines the connection between the Intercom Transceiver and the ICU.	
TxChannel	Type	Default Value
	uint64	0
	Description: Reflects the transmit channel set in the ICU.	
TxAudio	Type	Default Value
	boolean	False
	Description: Local audio connection to the Transceiver transmit audio stream.	
TxPower	Type	Default Value
	float32	0.0
	Description: Indicates the transmit power in dBm.	

15.5. ICU

Summary: The ICU component is used with the IntercomTransceiver to create a network intercom.

Description: The Network Intercom provides a simple method of running networked communications and is the easiest way to connect two operators to talk over a network. The ICU component is a control unit for the Transceiver components and removes most of the radio functionality such as bandwidth or propagation. Two intercoms must be in Intercom Mode (per the Net and Fill) and have a matching Channel, in the same way two radios must have a matching frequency.

The ICU helps to extend intercom capability beyond local operators and goes across a LAN or WAN allowing multiple Telestras to communicate with each other when full fidelity radios are not required or desired. The ICU also uses fewer credits than a network radio.

Note: The ICU component can be used with the Transceiver but that is not a recommended configuration.

Internal Parameters		
Channel	Type	Default Value
	uint64	0
	Description: Defines the ICU channel number. In order to receive over the network, both channel numbers must be equal. Valid values are 0-99,999.	

Control Inputs		
Fill	Type	Default Value
	fill	
	Description: Insert the fill created in the Comm Plan by double-clicking in the 'Value' column.	
Net	Type	Default Value
	uint32	1
	Description: Set in the Comm Plan. Set all nets to Intercom mode or use the default Clearcom net.	
TransceiverID	Type	Default Value
	id	unassigned
	Description: TransceiverID tells the ICU what IntercomTransceiver component it should control.	

15.6. MarkerTone

Summary: The MarkerTone component provides the ability to add navigational beacons to a model.

Description: The MarkerTone component provides the ability to add the Outer Marker, Middle Marker, Inner Marker, and Fan Marker navigational beacons to a model. The default values are set to FAA standards. While the component provides four separate beacons, the component can be considered only one of the beacons at a time. An input control selects which beacon to use.

Control Inputs		
Enable	Type	Default Value
	boolean	True
	Description: When TRUE MarkerTone is enabled.	
Frequency 1	Type	Default Value
	basic/float32	400
	Description: Sets the pitch for the keyed tone when number is set to 1. Default is set to 400Hz for the Outer Marker.	
Frequency 2	Type	Default Value
	basic/float32	1300
	Description: Sets the pitch for the keyed tone when number is set to 2. Default is set to 1300Hz for the Middle Marker.	
Frequency 3	Type	Default Value
	basic/float32	3000
	Description: Sets the pitch for the keyed tone when number is set to 3. Default is set to 3000Hz for the Inner Marker.	

Control Inputs		
Frequency 4	Type	Default Value
	basic/float32	3000
	Description: Sets the pitch for the keyed tone when number is set to 4. Default is set to 3000Hz for the Fan Marker.	
Frequency 5	Type	Default Value
	basic/float32	0
	Description: Sets the pitch for the keyed tone when number is set to 5.	
Frequency 6	Type	Default Value
	basic/float32	0
	Description: Sets the pitch for the keyed tone when number is set to 6.	
Frequency 7	Type	Default Value
	basic/float32	0
	Description: Sets the pitch for the keyed tone when number is set to 7.	
Frequency 8	Type	Default Value
	basic/float32	0
	Description: Sets the pitch for the keyed tone when number is set to 8.	
Gain	Type	Default Value
	basic/float32	1.0
	Description: Gain applies strength to the MarkerTone.	

Control Inputs		
Interval1	Type	Default Value
	basic/uint16	0
	Description: Used to set the delay between repeating the keyed word when number is set to 1. Units are in seconds.	
Interval2	Type	Default Value
	basic/uint16	1
	Description: Used to set the delay between repeating the keyed word when number is set to 2. Units are in seconds.	
Interval3	Type	Default Value
	basic/uint16	0
	Description: Used to set the delay between repeating the keyed word when number is set to 3. Units are in seconds.	
Interval4	Type	Default Value
	basic/uint16	1
	Description: Used to set the delay between repeating the keyed word when number is set to 4. Units are in seconds.	
Interval5	Type	Default Value
	basic/uint16	1
	Description: Used to set the delay between repeating the keyed word when number is set to 5. Units are in seconds.	

Control Inputs		
Interval6	Type	Default Value
	basic/uint16	1
	Description: Used to set the delay between repeating the keyed word when number is set to 6. Units are in seconds.	
Interval7	Type	Default Value
	basic/uint16	1
	Description: Used to set the delay between repeating the keyed word when number is set to 7. Units are in seconds.	
Interval8	Type	Default Value
	basic/uint16	1
	Description: Used to set the delay between repeating the keyed word when number is set to 8. Units are in seconds.	
IdentIndex	Type	Default Value
	basic/uint8	0
	Description: Selects which marker beacon (IdentChar or word) is to be transmitted. A value of 0 selects none. A value of 1 selects Ident1Char[0-3] (the Outer Marker). A value of 2 selects the second word (Middle Marker). A value of 3 selects the third word (Inner Marker). A Value of 4 selects the fourth word (Fan Marker).	

Control Inputs		
StrictTiming	Type	Default Value
	boolean	True
	Description: When True, MarkerTone uses proper MarkerTone inter-character timing which is a one (1) dot gap between ident characters and a 2 dot gap after the final character. When False, MarkerTone uses standard Morse code inter-character timing which is a three (3) dot gap between ident characters and a three (3) dot gap after the final character.	
Wordrate 1-2, 4-8	Type	Default Value
	basic/uint8	8.0
	Description: Determines the rate at which the word is keyed.	
Wordrate3	Type	Default Value
	basic/uint8	12.0
	Description: Determines the rate at which the word is keyed.	

Audio Output		
OutSignal	Type	Default Value
	audio	n/a
	Description: OutSignal is the output signal from the MarkerTone component, which may be connected to another component or directed to an output highway.	

Control Output		
Active	Type	Default Value
	boolean	False
	Description: When keyer is keying a dot or a dash this goes True, essentially mimicking the ‘beep’ of the keyer.	
Busy	Type	Default Value
	boolean	False
	Description: When OutSignal is Active Audio, Busy is True.	

Internal Parameters		
Ident1	Type	Default Value
	ident	T
	Description: The identifier played by the keyer when number is equal to 1. A value of T plays a single dash in Morse Code.	
Ident2	Type	Default Value
	ident	A
	Description: The identifier played by the keyer when number is equal to 1. A value of A plays a dot dash in Morse Code.	
Ident3	Type	Default Value
	ident	E
	Description: The identifier played by the keyer when number is equal to 1. A value of E plays a single dot in Morse Code.	

Internal Parameters		
Ident4	Type	Default Value
	ident	I
	Description: The identifier played by the keyer when number is equal to 1. A value of I plays a dot dot in Morse Code.	
Ident5-8	Type	Default Value
	ident	n/a
	Description: The identifier played by the keyer when number is equal to 1. These are useful if you want to create your own marker.	

15.7. MorseKeyer

Summary: The MorseKeyer component is used to create a 4-letter morse code sequence.

Description: The MorseKeyer component is used to translate 4 letter words into Morse Code. The component allows control over the wordrate, interval and the frequency (pitch) of the generated tone. In addition to the usual letters and numbers defined in Morse code, it also includes the characters '*' and '-' to represent individual dot and dash combinations.

Control Inputs		
Enable	Type	Default Value
	boolean	True
	Description: When True it enables the Morse Keyer. When False the Morse Keyer is disabled.	
Frequency	Type	Default Value
	float32	1.0
	Description: Provides the frequency (pitch) of the Morse tone. Units are in Hertz.	
Gain	Type	Default Value
	float32	1.0
	Description: Scales the strength of the generated tone.	
Ident	Type	Default Value
	ident	n/a
	Description: The 4-letter word that connects into Morse code. Ident is a special variable Type which is defined as 4 ASCII characters concatenated together. A HostIn component must be used to drive this parameter.	

Control Inputs		
Interval	Type	Default Value
	uint16	1
	Description: The total cycle in seconds before the ident message starts playing including the pause. i.e. The wordrate time plus the pause after the ident message (wordrate - interval) before a new cycle begins. Interval number must be larger than the wordrate.	
ILSPause	Type	Default Value
	boolean	False
	Description: When False, MorseKeyer uses normal Morse code inter-character timing which is a three (3) dot gap between all ident characters. When True, MorseKeyer uses a standard morse code inter-word timing between the first and second characters (i.e. a seven (7) dot gap). Inter-character timing for subsequent ident characters will use standard Morse code inter-character spacing (i.e. a three (3) dot gap).	
Wordrate	Type	Default Value
	uint8	1
	Description: Determines the rate at which the word is keyed.	

Audio Output		
OutSignal	Type	Default Value
	audio/audio	n/a
	Description: Outputs the keyed word when the PTT control is enabled.	

Control Output		
Active	Type	Default Value
	boolean	FALSE
	Description: When keyer is keying a dot or dash this goes to True, essentially mimicking the ‘beep’ of the keyer.	
Busy	Type	Default Value
	basic/boolean	FALSE
	Description: This flag indicates if the component is actively producing Morse audio.	

15.8. RCUbasic

Summary: This component paired with the Transceiver component provides radio modeling for the radio simulated environment.

Description: The most commonly configured radio uses two components, the Transceiver and RCUbasic. The Radio Control Unit (RCU) basic requires a CommPlan fill to access radio modes, waveforms, default settings, etc. If the control inputs are set from an external source (e.g. host control) they will override the default values in the fill. For example, RCUbasic has a loaded fill and is using Net 1, which defaults the frequency of the radio to 101MHz. This information is sent to the paired Transceiver that shares the same Transceiver ID as the RCUbasic. Suppose a user wanted to use the net default, except that they wanted to use a new frequency of 105MHz. By manually setting the Frequency of the RCUbasic to 105Mhz, it will override the net default, and the Transceiver will set its frequency to 105MHz.

In general, settings that relate to how the radio appears on the network are in the Transceiver, and settings which control the radio for simulation are placed in the RCUbasic.

Control Inputs		
Crypto Key	Type	Default Value
	uint16	0
	Description: If two radios are using encryption then they must have matching crypto keys in order to communicate.	
Frequency	Type	Default Value
	uint64	0
	Description: The current radio tune frequency in Hz.	

Control Inputs		
FreqHop HopSet-WOD	Type	Default Value
	uint16	0
	Description: Identifies the set of frequencies used in hopping pattern. The default of 0 = match all.	
FreqHop Lock-OutID	Type	Default Value
	uint16	0
	Description: Identifies the set of frequencies that are excluded from hopping pattern. The default of 0 = match all.	
FreqHop NetID	Type	Default Value
	uint16	0
	Description: Identifies the frequency hopping Net ID. A frequency hopping radio must have a non-zero Net ID to be considered actively hopping.	
FreqHop SyncTOD	Type	Default Value
	uint32	0
	Description: Identifies the frequency hopping Time of Day. The default of 0 = match all.	
FreqHop System	Type	Default Value
	uint16	0
	Description: This parameter is not used. ASTi recommends leaving this set to 0.	

Control Inputs		
FreqHop Tran- SecKey	Type	Default Value
	uint16	0
	Description: Identifies the frequency hopping transmission security key used in generating hopping patterns. The default of 0 = match all.	
Net	Type	Default Value
	uint32	1
	Description: Selects the Net from the Comm Plan fill. Net defines the core radio parameters including frequency, Tx frequency, waveform, crypto, and frequency hopping. The Net parameters are set in the Comm Plan.	
TxFre- quency	Type	Default Value
	uint64	0
	Description: The transmitter frequency of the Transceiver in Hz. For use only if the transmit frequency differs from the receive frequency.	
TxPower	Type	Default Value
	float32	0
	Description: The transmit power for the radio.	

Control Outputs		
FillName	Type	Default Value
	string	n/a
	Description: Reports what is entered in the Fill. Typically not connected to an output.	
NetIndi- cate	Type	Default Value
	uint32	0
	Description: Reports what is entered in the Transceiver ID. Typically not connected to an output.	

Internal Parameters		
Fill	Type	Default Value
	fill	n/a
	Description: Identifies the CommPlan fill to use with the Transceiver. Insert the fill created in the Comm Plan by double-clicking under the 'Value' column. A radio must have a fill in order to operate.	
Transceiv- erID	Type	Default Value
	id	UNASSIGNED
	Description: TransceiverID tells RCUBasic what Transceiver component it should control. The Transceiver must have a matching TransceiverID.	

15.9. Receiver

Summary: The Receiver component communicates with the Transceiver component to force receive capability.

Description: The Receiver component is an RCU included in the Radio group. It is designed to communicate with the Transceiver component to force receive capability only. It is a subset of the features and controls located inside the RCUBasic component.

The benefit of using the Receiver over the RCUBasic (or all other all-purpose RCUs) is that it charges a reduced number of credits for the reduced functionality. As a result, the component is useful for replicating Guard Receivers and Navigational Aides such as an ADF Receiver, or a TACAN Receiver.

For more information about using RCUs and the Transceiver, see the preface to the Radio group and the RCUBasic component.

Control Inputs		
CryptoKey	Type	Default Value
	uint16	0
	Description: If two radios are using encryption then they must have matching crypto keys for the crypto modes. This field must match in order for proper radio tuning.	
Frequency	Type	Default Value
	uint64	0
	Description: The current radio tune frequency in Hz.	

Control Inputs		
FreqHopNetId	Type	Default Value
	uint16	0
	Description: Identifies the frequency hopping network.	
FreqHopSyncTOD	Type	Default Value
	uint32	0
	Description: Identifies the Time of Day used in frequency hopping.	
FreqHop-Trans-SecKey	Type	Default Value
	uint16	0
	Description: Identifies the transmission security key used in generating hopping patterns.	
FreqHopSetWOD	Type	Default Value
	uint16	0
	Description: Identifies the set of frequencies used in hopping pattern.	
FreqHop-LockOutId	Type	Default Value
	uint16	0
	Description: Identifies the set of frequencies that are excluded from hopping pattern.	
Net	Type	Default Value
	uint32	1
	Description: Net defines the core radio features including frequency, Tx frequency, waveform, crypto, and frequency hopping. The Net parameters are set in the Comm Plan.	

Internal Parameters		
Fill	Type	Default Value
	fill	
	Description: Insert the fill created in the Comm Plan by double-clicking under the 'Value' column.	
TransceiverID	Type	Default Value
	id	UNASSIGNED
	Description: TransceiverID tells the RCU what Transceiver component it should control.	

15.10. Relay

Summary: The Relay component is used to link up any pairs of radios from a bank of eight radios.

Description: The Relay component connects up to eight Transceiver Components together in order to form radio relays. A radio relay will take all received audio from a radio and re-transmit out another radio. The Relay Component allows the user to make up to four pairs of relays across eight radios. In the most basic case, link the received audio from a transceiver to ReceiveAudio1 of the Relay Component. The RadioSelector1 field sets which radio that audio gets routed to, where 2 = TransmitAudio2, 3 = TransmitAudio3, etc.

In the example below, Transceiver 2 and 3 are set as a relay pair, and Transceiver 1 and 4 are also a pair. As in the real world, radios in Relay Mode should have different frequencies.

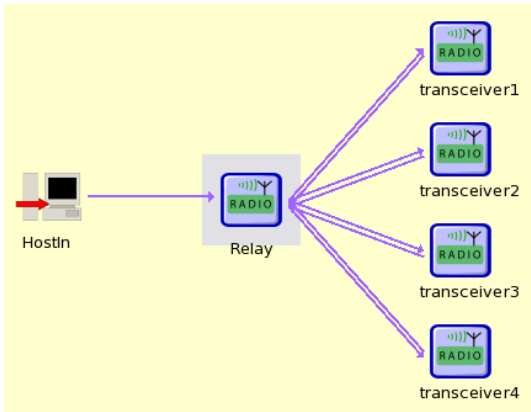


Figure 14: Basic Relay Model Example

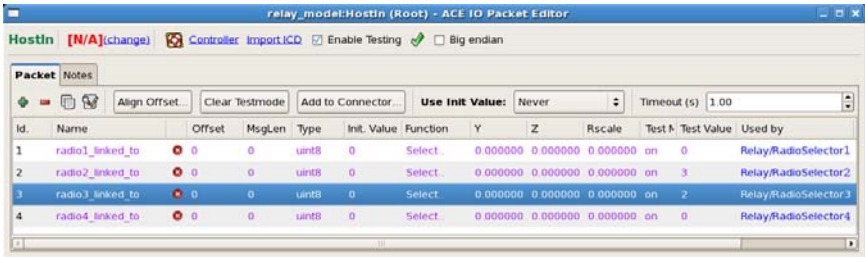


Figure 15: Relay Host Control

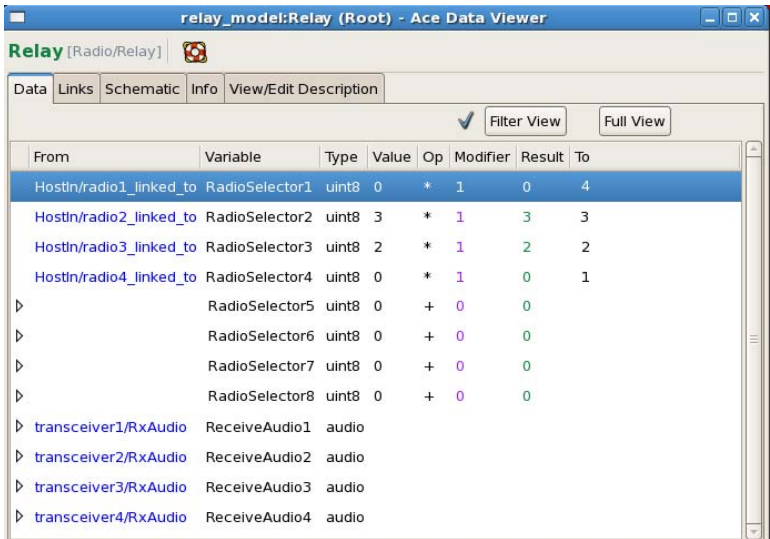


Figure 16: Relay Component

Audio Inputs		
ReceiveAudio1	Type	Default Value
	audio	n/a
	Description: The receive audio from the selected Transceiver routed into the component.	
ReceiveAudio2	Type	Default Value
	audio	n/a
	Description: The receive audio from the selected Transceiver routed into the component.	

Audio Inputs		
ReceiveAudio3	Type	Default Value
	audio	n/a
	Description: The receive audio from the selected Transceiver routed into the component.	
ReceiveAudio4	Type	Default Value
	audio	n/a
	Description: The receive audio from the selected Transceiver routed into the component.	

Control Inputs		
RadioSelector1	Type	Default Value
	uint8	0
	Description: RadioSelector1 chooses what Transmit Audio stream that ReceiveAudio1 will get routed to. A value of zero means ReceiveAudio1 is not relayed to any other radio.	
RadioSelector2	Type	Default Value
	uint8	0
	Description: RadioSelector2 chooses what Transmit Audio stream that ReceiveAudio2 will get routed to. A value of zero means ReceiveAudio2 is not relayed to any other radio.	

Control Inputs		
RadioSelector3	Type	Default Value
	uint8	0
	Description: RadioSelector3 chooses what Transmit Audio stream that ReceiveAudio3 will get routed to. A value of zero means ReceiveAudio3 is not relayed to any other radio.	
RadioSelector4	Type	Default Value
	uint8	0
	Description: RadioSelector4 chooses what Transmit Audio stream that ReceiveAudio4 will get routed to. A value of zero means ReceiveAudio4 is not relayed to any other radio.	
RadioSelector5	Type	Default Value
	uint8	0
	Description: RadioSelector5 chooses what Transmit Audio stream that ReceiveAudio5 will get routed to. A value of zero means ReceiveAudio5 is not relayed to any other radio.	
RadioSelector6	Type	Default Value
	uint8	0
	Description: RadioSelector6 chooses what Transmit Audio stream that ReceiveAudio6 will get routed to. A value of zero means ReceiveAudio6 is not relayed to any other radio.	
RadioSelector7	Type	Default Value
	uint8	0
	Description: RadioSelector7 chooses what Transmit Audio stream that ReceiveAudio7 will get routed to. A value of zero means ReceiveAudio7 is not relayed to any other radio.	

Control Inputs		
RadioSelector8	Type	Default Value
	uint8	0
	Description: RadioSelector8 chooses what Transmit Audio stream that ReceiveAudio8 will get routed to. A value of zero means ReceiveAudio8 is not relayed to any other radio.	

Audio Outputs		
Transmit Audio1	Type	Default Value
	audio	
	Description: A link to the transmit audio field of the selected Transceiver. Audio is routed from the Receive Audio lines based on the Radio Selectors.	
Transmit Audio2	Type	Default Value
	audio	
	Description: A link to the transmit audio field of the selected Transceiver. Audio is routed from the Receive Audio lines based on the Radio Selectors.	
Transmit Audio3	Type	Default Value
	audio	
	Description: A link to the transmit audio field of the selected Transceiver. Audio is routed from the Receive Audio lines based on the Radio Selectors.	

Audio Outputs		
Transmit Audio4	Type	Default Value
	audio	
	Description: A link to the transmit audio field of the selected Transceiver. Audio is routed from the Receive Audio lines based on the Radio Selectors.	
Transmit Audio5	Type	Default Value
	audio	
	Description: A link to the transmit audio field of the selected Transceiver. Audio is routed from the Receive Audio lines based on the Radio Selectors.	
Transmit Audio6	Type	Default Value
	audio	
	Description: A link to the transmit audio field of the selected Transceiver. Audio is routed from the Receive Audio lines based on the Radio Selectors.	
Transmit Audio7	Type	Default Value
	audio	
	Description: A link to the transmit audio field of the selected Transceiver. Audio is routed from the Receive Audio lines based on the Radio Selectors.	
Transmit Audio8	Type	Default Value
	audio	
	Description: A link to the transmit audio field of the selected Transceiver. Audio is routed from the Receive Audio lines based on the Radio Selectors.	

14.11. Satellite

Summary: The satellite component is a simulated satellite, which handles uplink-downlink relay of signals transmitted from a simulated radio in SATCOM mode, including simulation of mode-dependent delays.

Description: The satellite component represents a simulated satellite, and can be configured to relay uplink signals from radios configured in SATCOM mode.

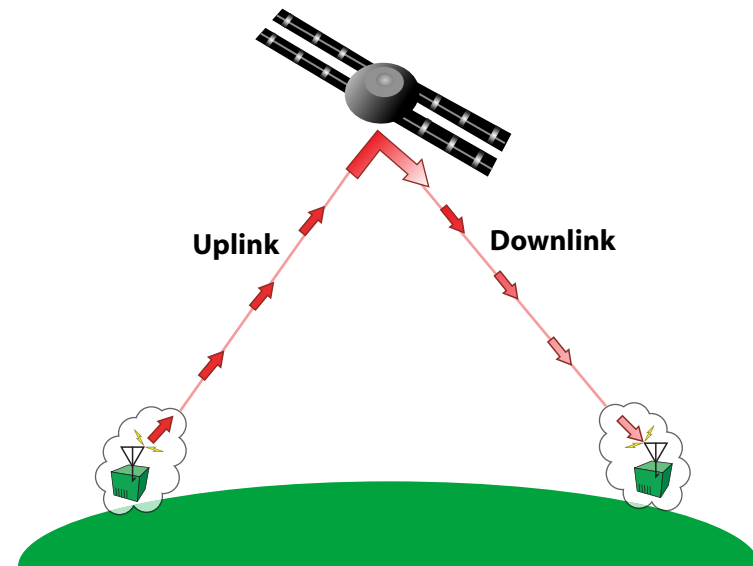
Realistic simulation of satellite parameters, including delays based on SATCOM sub-modes, world position, uplink and downlink bands, are included.

Add multiple satellite components to a model to simulate multiple, independent satellites.

SATCOM sub-modes include:

- 5k Dedicated
- 5k DASA
- 5k DAMA
- 25k Dedicated
- 25k DC DASA
- 25k AC DAMA
- 25k DC DAMA

Propagation effects including smooth earth occulting and terrain occulting (through use of a terrain server) can also be included.



Control Inputs		
Uplink-Frequency	Type	Default Value
	uint64	0
	Description: Base frequency of satellite downlink frequency band in Hertz.	
Downlink-Frequency	Type	Default Value
	uint64	0
	Description: Base frequency of satellite uplink frequency band in Hertz.	

Control Inputs		
Passband	Type	Default Value
	uint64	0
	Description: Defines the frequency range of the satellite uplink and downlink bands in Hertz.	
TxPower	Type	Default Value
	float32	1.0
	Description: The transmit power for the satellite downlink transmitter in Watts.	
Channels	Type	Default Value
	uint32	1
	Description: Number of simultaneous channels that can be supported by the satellite.	
Fixed-Delay	Type	Default Value
	uint32	0
	Description: Overrides mode based delays and forces a fixed delay value. Specified in ms.	
Delay5k	Type	Default Value
	uint32	0
	Description: Delay applied between when the satellite receives an uplink signal and when it transmits the downlink signal when in SATCOM 5k mode. Specified in ms.	

Control Inputs		
Delay5kD ASA	Type	Default Value
	uint32	0
	Description: Delay applied between when the satellite receives an uplink signal and when it transmits the down link signal when in SATCOM 5k DASA mode. Specified in ms.	
Delay5kD AMA	Type	Default Value
	uint32	0
	Description: Delay applied between when the satellite receives an uplink signal and when it transmits the down link signal when in SATCOM 5k DAMA mode. Specified in ms.	
Delay25k	Type	Default Value
	uint32	0
	Description: Delay applied between when the satellite receives an uplink signal and when it transmits the down link signal when in SATCOM 25k mode. Specified in ms.	
Delay25k DCDASA	Type	Default Value
	uint32	0
	Description: Delay applied between when the satellite receives an uplink signal and when it transmits the down link signal when in SATCOM 25k DCDASA mode. Specified in ms.	
Delay25k ACDAMA	Type	Default Value
	uint32	0
	Description: Delay applied between when the satellite receives an uplink signal and when it transmits the down link signal when in SATCOM 25k ACDAMA mode. Specified in ms.	

Control Inputs		
Delay25k DCDAMA	Type	Default Value
	uint32	0
	Description: Delay applied between when the satellite receives an uplink signal and when it transmits the down link signal when in SATCOM 25k DCDAMA mode. Specified in ms.	
Domain-NameIn	Type	Default Value
	string	n/a
	Description: Allows a host platform to configure the Domain name string remotely. This input is typically used in conjunction with the Control > NumtoString component. Refer to the DomainName string definition for syntax.	
ProtocolIDIn	Type	Default Value
	string	
	Description: Allows a host platform to configure the ProtocolID string remotely. This input is typically used in conjunction with the Control > NumtoString component. Refer to the Protocol ID string definition for syntax.	
ForceCenterOfEarth	Type	Default Value
	boolean	False
	Description: A manual override which places the Satellite at 0, 0, 0 geocentric world position. This disables all the propagation effects on transmit/receive.	

Internal Parameters		
Satellite-Name	Type	Default Value
	string	n/a
	Description: Name of the Satellite.	
Domain-Name	Type	Default Value
	string	n/a
	Description: The DomainName is part of the ASTiNet property set and defines a common communications environment. All radios within the same domain have the ability to communicate and those radios in separate domains can never communicate. It is analogous to a DIS Exercise ID or an HLA Federation name. When defined as an ASCII string the Transceiver is natively an ASTiNet radio. When that string matches a defined DIS domain (at the project level) it will map the domain name to a DIS exercise ID. As a shortcut, define a DIS Exercise ID by entering “DIS:N”, where N=1 through 255. For example, “DIS:1” would but the radio in DIS Exercise ID #1. In the satellite component, this specifies the domain in which the satellite will be active. Leaving this field blank will cause it to act as a wild card and satellite will be active in all domains.	

Internal Parameters		
Proto- colID	Type	Default Value
	string	n/a
	Description: Optional configuration for when you have a non-ASTiNet radio. DIS is currently the only supported protocol. The syntax for setting up a DIS radio is the string format of 'DIS:site.app.entity.radio' or 'DIS:entity.radio' or 'DIS:radio'. If the site and app are excluded from the string, the radio environment will assign the default site and app from the DIS file in the Domain Editor. If entity is excluded, the radio environment will assign a random/unique number entity ID for the radio. This can also be set in the Radio Helper. This must be unique within the DIS network. For example: DIS:100.100.1.1 will set: Site=100, App=100, Entity=1, and Radio=1 In the Satellite component, the protocol ID affects the DIS ID of the temporary transmitters setup to retransmit the signal.	
WorldPo- sitionBus	Type	Default Value
	id	unassigned
	Description: Optional configuration which allows for a Satellite to be assigned to a world position bus. If undefined, the Satellite will default to X,Y,Z = 0,0,0. Alternatively if an entity exists on the DIS network that matches the Site.App.Entity ID within the Protocol ID field of the Satellite, the Satellite will inherit that entity's world position information. When a Bus name is defined the Satellite will be located (X,Y,Z) based on the Platform position component which is tied to the same Bus name used in conjunction with the Platform/GeocentricPosition or GeodeticPosition.	

14.12. Transceiver

Summary: The Transceiver works in conjunction with an RCU component to create a simulated radio.

Description: The Transceiver is, as the name suggests, a component used to model radio transmission and radio reception. Used in conjunction with a Radio Control Unit (RCU), an Intercom Control Unit (ICU), a Receiver or a Transmitter component it forms the basis of a Radio, Network Intercom, Receiver or Transmitter respectively.

Some of the core features of the Transceiver include:

- Host Control of core radio parameters
- Modulation Matching (AM, FM, USB, LSB, etc.)
- RF Propagation Modeling based on world position, frequency, etc.
- Crypto state and sound modeling
- Frequency Hopping modeling
- Jamming support
- Squelch control
- Terrain support (in conjunction with an external Terrain Server)
- Antenna Gain, Cable Loss, RXTuneTone and other advanced radio parameters
- Audio and Tactical Data Link support
- Multi-protocol support including local, ASTiNet, DIS and HLA

The Transceiver provides a generic, high-level radio simulation, which includes transmit and receive operations, frequency tuning effects, AM and FM modulation modes, signal-strength variation due to range, transmit power, antenna and receiver gain, RF and internal noise, and propagation path loss, sidetone signal return, and support for crypto and frequency hop behaviors, and more.

At the simplest level, the Transceiver component provides a simulation of the interface between the signal and data flow within a radio, and the simulated radio frequency (RF) environment. Just as in the real world, the signals passed between a real transceiver and the remainder of the radio sub-system provide information to/from the Transceiver, and include the transmit and receive audio signals (voice/tones), transmit and receive data message signals (including Link-16, IDM, ACARS, etc), and control signals that determine the behavior of the Transceiver (such as tuned frequency, modulation mode, bandwidth, etc).

In effect the Transceiver may be regarded as having two bi-directional information interfaces, and a single (mostly) one-way control interface. One of the bi-directional information interfaces operates internally, within the simulated radio and/or vehicle, and operates at base-band, while the other bi-directional interface operates in the RF bands, and provides wireless transmission and reception. The control interface determines the translation between the base-band and RF environments.

A Transceiver component *must* be used with an RCU object to form an operational “radio”, with the RCU providing the greater portion of the control interface data values. Applicable RCU sub-class objects that may be used with the Transceiver include: RCU, ICU, transmitter, and receiver components respectively. The nature of the selected RCU will determine the available capabilities of the Transceiver. For example, selection of a ‘receiver’ RCU object will limit the operation of the Transceiver to the reception of RF originated signals.

The Transceiver can be used to implement various fidelity levels of RF modeling from simple frequency matching, through full-fidelity simulation of specific radio types, including propagation and ranging, bandwidth overlap, antenna gain, etc. Ranging behavior requires that each radio have a ‘world position’ describing the location of the radio on or above the Earth’s surface. If a radio is configured to operate using the DIS networking protocol, another option available is to attach a radio to an externally simulated entity, using the Entity Attach feature. Once attached, the Transceiver position will be slaved to the selected entity and all range calculations will be based on the supplied entity position.

The received signal strength is computed for all in-tune radios based on the power of the transmitter, the antenna gains of the transmitter and receivers, and the relative world positions. If the terrain interface is installed, the gain factor for the in-tune radios is factored in the calculator. If frequency hopping or encryption is enabled, the parameters of the transmitter and receiver are compared to see if the audio is received. (In frequency hopping mode, the frequency field is ignored. The frequency is implied in the selected Net ID hopset). If multiple transmitters are broadcasting on the same frequency, the Transceiver will typically do one of two things. For AM signals, the received RF power is combined and the received audio is a sum of the transmitted signals in proportion to their signal strength. For FM signals, only the strongest received signal is included.

Once the received power is determined, the RF SNR is calculated. The noise level is determined by thermal noise, internal radio noise, and other parameters, which are set in the Transceiver. The RF SNR is then compared to the squelch level. If the ratio is less than the squelch level, the signal is not received. Setting the squelch to zero disables the squelch.

After determining that the signal is received, the signal power and noise power are affected by the AGC. Additionally, when the squelch is off, the maximum AGC will determine the background noise when no signal is received. The received audio is routed out of the component locally and/or onto the Intercom Bus Service where an operator can hear it (typically thru a communications panel component).

The Transceiver can transmit as well as receive. When the radio transmits the reception is cut off (assuming half-duplex operation).

To support crypto, simulation radios use a library of crypto tones (soundfiles) with each Transceiver. This library greatly simplifies the encrypted radio simulation by automatically playing tones (such as a preamble, a postamble, or a mismatch tone) at the appropriate times during a secure radio transmission or reception. For more information on crypto sound files see the ASTi Support page specifically www.asti-usa.com/support/faq/telestra4/11.html.

Radio jamming occurs when a receiving radio operating in say FM, is blocked by a strong transmission, most often using a non-matching modulation mode (e.g. PULSE), causing any desired signal to be masked by the unwanted "jammer". This capability is supported in the Transceiver.

The jammed status of a receiver is determined automatically by the receiver. The audio associated with a receiver being jammed is implemented through the Transceiver component utilizing the soundfile library system to organize the jamming sounds. Each Transceiver may be set-up to point at a particular Jamming Library (common or otherwise), and Jamming Group. The soundfile indexes populated within the sound library/group map to possible modes of the receiver object.

The receiving radio automatically chooses which soundfile to play when it is jammed. The soundfile (index) chosen is determined by the receiver mode. The table below specifies what sound index is played based on the mode.

Receiver Mode	Soundfile Index Played
FM	2
AM	3
SATCOM	4
CW	8
USB	9
LSB	10
Pulse	11
SSBF	13

If the terrain interface is configured, the radio environment will determine in-tune transmitter-receiver pairs and will generate data packets containing the transmitter-receiver world positions. These packets may be processed by the host computing system combined with a suitable terrain database to determine highly accurate line-of-sight terrain obscuration checks in addition to the range calculations. Without the terrain package, ranging is limited by a calculation based upon a WGS-84 model of the Earth's curvature.

Notes:

1. For frequencies between 1 and 100,000, no background noise or signal attenuation effects are simulated. These frequencies provide a clear channel of communication, regardless of transmission power, world position, etc. The frequencies are typically reserved for network intercom communications (i.e. INTERCOM mode)
2. A Transceiver will default to a world position of geocentric X, Y, Z = 0, 0, 0 (that is, the center of the earth). At this position a radio will receive any radios using the same frequency without any signal loss or occulting. This feature can be used to model a radio that monitors a particular radio band, without regard to position or transmit power.

Audio Inputs		
External-Noise	Type	Default Value
	audio	n/a
	Description: Attach an input signal to replace the NoiseSource. Overrides the white noise generator if signal is attached and active.	
RxTune-Tone	Type	Default Value
	audio	n/a
	Description: Local audio connection that will be mixed into the receive audio path of the radio.	
TxAudio	Type	Default Value
	audio	n/a
	Description: Local audio connection to the Transceiver transmit audio stream.	

Control Inputs		
AntennaGain	Type	Default Value
	float32	1.0
	Description: The linear antenna gain applied to the radio receive signal.	
Cableloss	Type	Default Value
	float32	1.0
	Description: The loss factor representing the antenna cable, a value of 1.0 represents no loss.	

Control Inputs		
BFOGain	Type	Default Value
	float32	1.0
	Description: The gain associated with the BFO audio.	
BFOFrequency	Type	Default Value
	float32	0.0
	Description: Beat Frequency Oscillator (BFO) in the Transceiver. The BFO has a tone strength that is proportional to the received carrier strength, generally used for detecting the Morse code keying present on a continuous wave beacon.	
CryptoGroup	Type	Default Value
	float32	1
	Description: Allows for host control between different crypto groups as defined in the sound library. For example, a user may change from a KY-28 to a KY-58.	
CryptoGain	Type	Default Value
	float32	1
	Description: The gain level of the crypto sound.	
CryptoEnable	Type	Default Value
	boolean	True
	Description: Allows a user to disable crypto even when crypto parameters are enabled in the Transceivers associated RCU.	

Control Inputs		
Crypto Only	Type	Default Value
	boolean	False
	Description: When set to True and if the radio is in crypto mode (system and key are both non-zero), then the radio will not receive clear transmissions and will not play the RX_Clear playsound. If set to True and the radio is in clear mode (system or key are 0) then the radio does receive clear transmissions.	
Domain-NameIn	Type	Default Value
	string	n/a
	Description: Allows a host platform to configure the Domain name string remotely. This input is typically used in conjunction with the Control > NumtoString component. Refer to the DomainName string definition for syntax.	
ForceCenterOfEarth	Type	Default Value
	boolean	False
	Description: A manual override which places the Transceiver at 0, 0, 0 geocentric world position. When true the radio is located at the center of the earth (0, 0, 0). Set to the center of the earth to not have any propagation effects.	
Interference Library	Type	Default Value
	playsound_library	0
	Description: The playsound library with the playsound that is played when the radio senses it is being jammed. See the Transceiver description for more information on jamming including a table for receiver modes and soundfile index.	

Control Inputs		
Interference Group	Type	Default Value
	playsound_group	1
	Description: The playsound group with the playsound that is played when the radio senses it is being jammed.	
Interference Gain	Type	Default Value
	float32	1.0
	Description: The gain applied to jamming audio.	
NoiseGain	Type	Default Value
	float32	.1
	Description: Gain associated with the internal Transceiver noise.	
PowerIn	Type	Default Value
	boolean	True
	Description: Enable or disable power for the Transceiver. If not connected in the “From” field, default value is True.	
ProtocolIDIn	Type	Default Value
	string	n/a
	Description: Allows a host platform to configure the ProtocolID string remotely. This input is typically used in conjunction with the Control > NumtoString component. Refer to the ProtocolID string definition for syntax.	

Control Inputs		
ReceiveGain	Type	Default Value
	float32	1.0
	Description: Gain applied to the received audio. Equivalent to a volume knob.	
RxALC Enable	Type	Default Value
	boolean	False
	Description: This flag enables auto level control (ALC) on the received signal of the Transceiver. Loud signals are reduced in volume and quiet signals are boosted. This means other ASTi networked radios are received at a consistent volume, regardless of what system the radios are running on (DACS Model Builder, Teles- tra 3 Model Builder Visual, or T4 ACE).	
RxDataThreshold	Type	Default Value
	float32	.200
	Description: Equivalent to a squelch level but for reception of data.	
RxEnable	Type	Default Value
	Boolean	True
	Description: If True then receive is enabled for the Transceiver.	
SideFx Enable	Type	Default Value
	boolean	True
	Description: If true, then any applied voicing effects will be heard in the sidetone that is returned from this transceiver. Note that voicing effects are enabled through the Commplan in the waveform.	

Control Inputs		
Squelch-Level	Type	Default Value
	float32	.200
	Description: When the received Signal to Noise Ratio (SNR) is less than the squelch value given in this field, the gain is set to zero, providing the normal background noise suppression. To disable the squelch set the level to zero. To calculate the squelch level in dB multiply the squelch result displayed in the component by 20. This is useful when you want to compare the squelch level to signal level in dB.	
Squelch-Tail	Type	Default Value
	uint32	75
	Description: The duration of the “squelch tail” in milliseconds.	
Transmit AudioGain	Type	Default Value
	float32	1.0
	Description: Gain that is applied to the transmit signal of the radio before the signal enters the radio environment.	
TxALC Enable	Type	Default Value
	boolean	False
	Description: This flag enables the Auto Level Control (ALC) on the transmit signal of the Transceiver. This attempts to maintain a consistent and specified audio volume, reducing volume in loud signals and boosting the volume for quiet signals. This means other ASTi networked radio systems (whether its DACS Model Builder, Teles- tra 3 Model Builder Visual, or T4 ACE) will hear this radio at a consistent volume.	

Control Inputs		
TxEnable	Type	Default Value
	boolean	True
	Description: Defaults to True. If True then transmit is enabled for the Transceiver.	

Audio Outputs		
RxAudio	Type	Default Value
	audio	n/a
	Description: The Transceiver local audio receive audio stream, i.e. this is the receiving audio the Transceiver will pickup from in-tune transmitting radios.	
TxCrypto-Audio	Type	Default Value
	audio	n/a
	Description: The audio the crypto tones player generates for transmit states only.	

Control Outputs		
Jammed	Type	Default Value
	boolean	False
	Description: Indicates whether the Transceiver is being jammed or not.	

Control Outputs		
Range	Type	Default Value
	float32	0.0
	Description: The distance to the transmitter currently being received in meters.	
RxActive	Type	Default Value
	boolean	False
	Description: Indicates whether the Transceiver is receiving.	
RxAudio-Active	Type	Default Value
	boolean	False
	Description: When True the Transceiver receive output is non-zero. True if squelch is broken or audio is received.	
RxCrypt	Type	Default Value
	boolean	False
	Description: True when the radio is actively receiving a secure transmission.	
RxData-Dropped	Type	Default Value
	int32	0
	Description: The number of data packets that the radio detected coming from an intune transmitting radio when RxDataThreshold-Met was False and therefore were not forwarded to the local host.	

Control Outputs		
RxData-Forwarded	Type	Default Value
	int32	0
	Description: The number of data packets that were successfully received from an intune transmitting radio while RxDataThreshold-Met is True and that were therefore also forwarded to the local host.	
RxDataThreshold-Met	Type	Default Value
	boolean	False
	Description: True when the receiver is intune with a transmitter and the SNR (in dB) of the path from that transmitter to the receiver is at or above the threshold set using the RxDataThreshold value.	
RxFreq	Type	Default Value
	uint64	0
	Description: Transceiver receive frequency.	
RxPath-Factor	Type	Default Value
	float32	0.0
	Description: Pathloss factor associated with the transmission path, could come from an external terrain server or pathloss server, internal calculations or path factor. A value of 1.0 represents no loss.	
RxPower	Type	Default Value
	float32	-270.0
	Description: Indicates the receive power in dBm.	

Control Outputs		
RxTx LocationX	Type	Default Value
	Float64	0.0
	Description: When the radio receives it populates the transmitters location. This is the X position in a geocentric coordinate system of the radio that the transceiver is currently receiving from.	
RxTx LocationY	Type	Default Value
	Float64	0.0
	Description: When the radio receives it populates the transmitters location. This is the Y position in a geocentric coordinate system of the radio that the transceiver is currently receiving from.	
RxTx LocationZ	Type	Default Value
	Float64	0.0
	Description: When the radio receives it populates the transmitters location. This is the X position in a geocentric coordinate system of the radio that the transceiver is currently receiving from.	
RxVoice-TransmissionActive	Type	Default Value
	boolean	False
	Description: When True the Transceiver is receiving audio from another radio.	
SNR	Type	Default Value
	float32	-270.0
	Description: The Signal to Noise Ratio (SNR) of the received signal.	

Control Outputs		
TxActive	Type	Default Value
	boolean	False
	Description: Indicates whether the Transceiver is transmitting.	
TxVoice	Type	Default Value
	boolean	False
	Description: Indicates when the Transceiver is transmitting audio.	
TxTDL	Type	Default Value
	boolean	False
	Description: Indicates when the Transceiver is transmitting data.	
TxCrypt	Type	Default Value
	boolean	False
	Description: Indicates when the Transceiver is in crypto mode and actively transmitting.	
TxFreq	Type	Default Value
	uint64	0
	Description: Indicates the Transceiver transmit frequency.	

Internal Parameters		
Domain-Name	Type	Default Value
	string	blank
	Description: The DomainName is part of the ASTiNet property set and defines a common communications environment. All transceivers within the same domain have the ability to communicate and Transceivers in separate domains can never communicate. It is analogous to a DIS Exercise ID or an HLA Federation name. When defined as an ASCII string the Transceiver is natively an ASTiNet Radio. When that string matches a defined DIS domain (at the project level) it will map the domain name to a DIS exercise ID. As a shortcut, define a DIS Exercise ID by entering “DIS:N”, where N = 1 thru 255. For example, “DIS:1” would put the Transceiver in DIS Exercise ID 1. If this shortcut is used, you must ensure that the Exercise ID is not already in use by any DIS domain created in the Domain Editor. Important: This parameter is required.	
PowerBus	Type	Default Value
	id	UNASSIGNED
	Description: Connects to the Power Service and is used in conjunction with Power Service to receive power instead of using a control.	

Internal Parameters		
ProtocolID	Type	Default Value
	string	n/a
	Description: The Protocol ID field has a variety of use cases. It allows a radio to become a DIS radio by setting the DIS identifiers such as the Host ID and Radio ID. The ProtocolID also allows the Marking Field to be set, or for the radio to be considered "local" and not published on the network. DIS:### - Sets the Site, Application, Entity, and Radio IDs DIS:## - Sets the Entity and Radio ID DIS:# - Sets the Radio ID DIS: - Sets all four IDs automatically LCL: - Sets the radio to be Local and not published on the network. If any of the above numbers are not set, the remaining Identifiers are generated automatically. The Site and Application ID are set via the last two octets of the DIS IP Address. The Entity and Radio IDs are generated randomly. The above examples can be appended with another colon and the DIS Marking Field. For example, DIS:100.3:RedForce1, sets the Entity ID to 100, the Radio ID to 3, and the Marking Field to RedForce1. The Site and Application IDs will be set from the IP address.	
RadioBus	Type	Default Value
	id	UNASSIGNED
	Description: Assigns the Transceiver to the Intercom Bus Service. This is most commonly used in conjunction with the CommPanel which also uses the Intercom Bus Service. The Intercom Bus Service allows for input audio, sidetone audio, and output audio to be passed around among components that are connected to the bus.	

Internal Parameters		
Radio-Name	Type	Default Value
	string	blank
	Description: The radio name to describe what the Transceiver is modeling. For example, UHF_Radio1. Important: This parameter is required.	
RxState	Type	Default Value
	CryptoStateRx	RxCryptoOff
	Description: This reports the current crypto receive state. Possible states include: • RxCryptoOff • RxCryptoPreamble1 • RxCryptoPreamble2 • RxCryptoPostAmble • RxCryptoClear • RxCryptoMatch • RxCryptoMismatch • RxCryptoMismatchPreamble Each one corresponds to a different sound file that is played from the crypto sound library. Crypto receive states include: <pre> 0 // play no sound 1 // play match preamble 1 2 // play match preamble 2 3 // play match post amble 4 // play clear tone 5 // play match tone 6 // play mismatch tone 7 // play mismatch tone </pre>	

Internal Parameters		
TransceiverID	Type	Default Value
	id	UNASSIGNED
	Description: The connection to the radio control service. This bus defines the connection between the Transceiver and its RCU/ICU. Important: This parameter is required.	
TxState	Type	Default Value
	CryptoStateTx	TxCryptoOff
	Description: This reports the current crypto transmit state. Possible states include: - TxCryptoOff - TxCryptoPreamble1 - TxCryptoPreamble2 - TxCryptoPostAmble - TxCryptoClear - TxCryptoCrypt Each one corresponds to a different sound file that is played from the crypto sound library. Crypto transmit states include: 0, // play no sound 11 // play crypto preamble 1 12 // play crypto preamble 2 13 // play crypto post amble 14 // play clear tone 15 // play crypto tone	

Internal Parameters		
WorldPositionBus	Type	Default Value
	id	UNASSIGNED
	Description: Optional configuration which allows for a Transceiver to be assigned to a world position bus. If undefined, the Transceiver will default to X,Y,Z = 0,0,0. Alternatively if an entity exists on the DIS network that matches the Site.App.Entity ID within the Protocol ID field of the Transceiver, the Transceiver will inherit that entity's world position information. When a Bus name is defined the Transceiver will be located (X,Y,Z) based on the Platform position component which is tied to the same Bus name used in conjunction with the Platform/GeocentricPosition or Platform/GeodeticPosition.	

Internal Display		
Mode	Type	Default Value
	TunerMode	n/a
	Description: Displays of the current transceiver tuner mode (for example AM, FM, USB, LSB, etc.).	
Power-State	Type	Default Value
	boolean	True
	Description: Displays the power state of the Transceiver.	
RxFrequency	Type	Default Value
	uint64	0
	Description: Displays the current Transceiver reception frequency.	

Internal Display		
TxFrequency	Type	Default Value
	uint64	n/a
	Description: Displays the current Transceiver transmission frequency.	

14.13. Transmitter

Summary: This component is used to create a radio that is Transmit Only.

Description: The Transmitter component is an RCU included in the Radio Group. It is designed to communicate with the Transceiver component to force transmit capability only. It is a subset of the features and controls located inside RCUBasic.

The benefit of using the Transmitter over the RCUBasic (or other all-purpose RCUs) is that it charges a reduced number of credits for the reduced functionality. As a result, the component is useful for replicating Navigational Aids, such as VOR or NDB Transmitters.

For more information about using RCUs and the Transceiver, see the preface to the Radio Group and the RCUBasic component.

Control Inputs		
CryptoKey	Type	Default Value
	uint16	0
	Description: If two radios are using encryption then they must have matching crypto keys for the crypto modes. This field must match in order for proper radio tuning.	
FreqHopNetID	Type	Default Value
	uint16	0
	Description: Identifies the frequency hopping Net ID. A frequency hopping radio must have a non-zero Net ID to be considered actively hopping.	
FreqHopSyncTOD	Type	Default Value
	uint32	0
	Description: Identifies the Time of Day used in frequency hopping.	

Control Inputs		
FreqHop-Trans-SecKey	Type	Default Value
	uint16	0
	Description: Identifies the transmission security key used in generating hopping patterns.	
FreqHopHopSet-WOD	Type	Default Value
	uint16	0
	Description: Identifies the set of frequencies used in hopping pattern.	
FreqHop-Lock-OutID	Type	Default Value
	uint16	0
	Description: Identifies the set of frequencies that are excluded from hopping pattern.	
Frequency	Type	Default Value
	uint64	0
	Description: The current transmitter tune frequency in Hz.	
Net	Type	Default Value
	uint32	1
	Description: Net defines the core radio features including frequency, Tx frequency, waveform, crypto, and frequency hopping. The Net parameters are set in the Comm Plan.	
TxPower	Type	Default Value
	float32	1.0
	Description: Sets the transmit power for the Transmitter.	

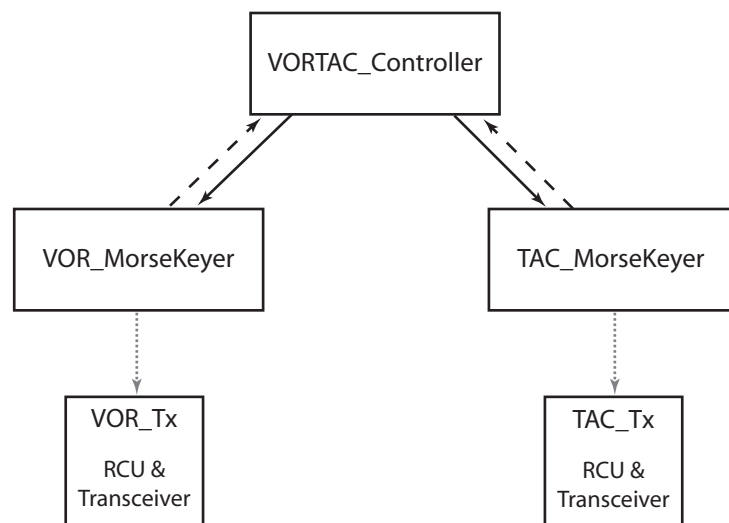
Internal Parameters		
Fill	Type	Default Value
	fill	n/a
	Description: Insert the fill created in the Comm Plan by double-clicking under the 'Value' column.	
TransceiverID	Type	Default Value
	id	UNASSIGNED
	Description: TransceiverID tells the Transmitter what Transceiver component it should control.	

14.14. VORTAC_Controller

Summary: The VORTAC Controller component is designed to control embedded identifier tone elements to simulate VOR and TACAN radios.

Description: The VORTAC Controller handles the timing of the VOR and TACAN tones to prevent overlap. Seven components are needed to implement a complete VORTAC simulation in a model. The TACAN and VOR radios each consist of a Transceiver, RCU, and MorseKeyer. Both MorseKeyers are driven by one VORTAC Controller. The VORTAC Controller will control (connect to) the Enable variable of the VOR and TACAN Morsekeyers to keep proper timing of the morse code transmission.

The MorseKeyer reports a busy flag to the VORTAC Controller to ensure accurate timing of the morse code transmissions. The busy variable of both MorseKeyers must be connected to the VOR_Busy and TACAN_Busy variables in the VORTAC Controller. (In some versions of ACE Studio, the VOR_Busy and TACAN_Busy variables are only visible in the component's "Full View.")



————> Ident, Intervals, Wordrate, Enables

- - -> Busy

.....> Morse Tones

Control Inputs		
Enable	Type	Default Value
	boolean	True
	Description: When True VORTAC_Controller is enabled.	
Ident	Type	Default Value
	ident	n/a
	Description: The ASCII characters that drive the MorseKeyer. Ident is a special variable Type which is defined as ASCII characters concatenated together. A HostIn component must be used to drive this parameter.	
Interval	Type	Default Value
	uint16	1
	Description: Used to set the delay between words. Units are in seconds.	
TACAN_Busy	Type	Default Value
	boolean	True
	Description: Connects to the TACAN MorseKeyer enable.	
VOR_Busy	Type	Default Value
	boolean	True
	Description: Connects to the VOR MorseKeyer enable.	

Control Inputs		
VOR_Count	Type	Default Value
	uint16	1
	Description: Number of keyed identifiers to skip before keying TACAN identifier. The VOR and TACAN identifier tones are mutually exclusive.	
Wordrate	Type	Default Value
	uint8	1
	Description: Determines the rate at which the word is played. Units are in dots per second. The faster the rate the higher the number.	

Control Outputs		
Vor_Enable	Type	Default Value
	boolean	False
	Description: Connects to the VOR MorseKeyer enable.	
Wordrate-Out	Type	Default Value
	uint8	1
	Description: Drives the VOR and TACAN MorseKeyer wordrates.	

Control Outputs		
IdentOut	Type	Default Value
	ident	n/a
	Description: Drives the VOR and TACAN MorseKeyer ident.	
IntervalOut	Type	Default Value
	uint16	1
	Description: Drives the VOR and TACAN MorseKeyer intervals.	
TACAN_Enable	Type	Default Value
	boolean	True
	Description: Connects to the TACAN MorseKeyer enable.	

15.0. Speech

15.1. SpeechFeed

Summary: Streams audio into the speech recognition engine.

Description: Link an audio stream containing an operator's speech to the SpeechFeed to perform speech recognition on the audio. Once a stream ID is created and selected the audio is fed into the speech recognition engine and processed according to the speech recognition configuration and grammars in the Project's SR Plans directory.

Consult the ACE Studio Technical User Guide (DOC-01-TELAS-UG-4) on how to configure speech recognition streams and grammars.

Audio Inputs		
AudioIn	Type	Default Value
	audio	n/a
	Description: The audio stream that is directed to the speech recognition engine.	

Internal Parameters		
StreamID	Type	Default Value
	ID	Unassigned
	Description: Name or identifier given to the speech recognition stream. Double-click "unassigned" to open the speech service window and then click "New Bus" to add a speech recognition stream. Click "set value" to set the stream in the component.	

15.2. TextToSpeech

Summary: Attaches to the ID of the specified stream and outputs the audio.

Description: The TexttoSpeech component selects up to 4 streams, or however many your license will allow, the gain is applied to the selected stream and the audio is routed out of the component.

Audio Outputs		
AudioOut	Type	Default Value
	audio	n/a
	Description: The audio stream out signal.	

Control Outputs		
Gain	Type	Default Value
	float32	0.0
	Description: The strength of volume applied to the audio outsignal.	
Active-Playing	Type	Default Value
	boolean	False
	Description: When true the component is active.	

Internal Parameters		
StreamID	Type	Default Value
	uint8	0
	Description: The ID of the audio streams, accepts multiple audio streams.	

16.0. Remote Control

16.1. URC-200

Summary: Interfaces with a single URC-200 live radio through an ACE-RIU channel.

Description: This component provides a low level interface to an URC-200 radio, through an ACE-RIU serial port.

The interface presented in the data viewer is similar to a combination Radio/RCU + Radio/Transceiver data viewer. Essentially this means there is a 'control' section and 'status' section to the component.

The 'control' section accepts user input, and drives the radio settings when applicable. Radio items in the 'control' section will generally be prefixed with a **RCU** string, as in **RCU_Preset**. Items in the 'status' section will not have a prefix string, as in **Preset**.

If the live radio fails to respond to a query for a specific status item (for example, a query for what its current preset is) the component will display an error indicator, like the number -1 or the string 'Unavailable'.

Input to the items in the 'control' section are naively checked for validity. One should not expect the component to know what frequency ranges are valid during live radio operation, for example. On the contrary, the component assumes the user completely understands the constraints on parameters for different modes of operation. If the user does input an invalid input, the transceiver will most likely return a NAK, which will eventually be flagged as an error in the **ErrorMask**.

For a more intuitive interface, refer to the Remote Control feature of RMS. The ideal interface to use with this component is the Live Radio Remote Control section of RMS.

There is also a Python API for interfacing with the Remote Control components in a model; this should allow programmatic control over live radio entities present in the system. Contact ASTi for more information.

The command list below is the complete list of commands sent to the live radio RS232 serial interface, from the component.

Command List

Command Name	Command Code
Zap	Z
Set Squelch	\$
Set Preset	P
Set Frequency	R
Set Transmit Frequency	T
Set Modulation Mode	M
Set Tx Modulation Mode	N
Set Text Mode	X
Store Preset	Q
Set Power Level	#
Set Beacon Mode	*
Set Keypad Control	+
Set Transmit Mode	B
Set Receive Mode	E
Synth Lock/Unlock	?01
Receive Sig Strength	?03
SW Version	?08
Squelch Level Setting	?09
Current Preset Status	?10
Text Mode Status	?11
General Status	?11
General Mode Status	?12
Squelch Status	?13

* Refer to the URC-200 operator manual for specifics on these commands.

Control Inputs		
Control Enable	Type	Default Value
	boolean	True
	Description: When TRUE the remote control function is enabled and control messages should be passing between the ACE-RIU and the live radio.	
Load Preset	Type	Default Value
	boolean	True
	Description: When TRUE the component will only attempt to load presets and will not attempt to modify preset settings. Preset settings are Frequency, Tx Frequency, Modulation Mode, Tx Modulation Mode, Power Level, and Text Mode. As a consequence of this, the 'Result' column of these settings (in the RCU section) will be overridden by the live radio's current setting.	
Store Preset	Type	Default Value
	boolean	False
	Description: Triggers a 'Store Preset' command on the remote radio, saving the current preset's state to the radio's internal memory. Refer to the radio's operator manual for more information. The save command will be sent when this control is initially set to TRUE from a FALSE state.	
External PTT	Type	Default Value
	boolean	False
	Description: When TRUE the control message exchange between the ACE-RIU and live radio are halted. It might be useful to have this TRUE when the live radio is transmitting, so that the radio is not simultaneously transmitting and processing remote control messages, as this might cause excessive noise on the transmitted signal.	

Control Inputs		
RCU_Preset	Type	Default Value
	uint8	0
	Description: Set the live radio preset to switch to. Valid presets are in the range of 0-9.	
RCU_Freq	Type	Default Value
	uint64	0
	Description: Set the live radio frequency to tune to. Valid frequencies will vary depending on other factors; refer to the radio's operator manual for more information.	
RCU_TxFreq	Type	Default Value
	uint64	0
	Description: Set the live radio transmit frequency to tune to. The component will not handle setting the frequency and transmit frequency in a coupled manner; that is, if one requires the two frequencies to be the same he must set both of them separately. Valid frequencies will vary depending on other factors; refer to the radio's operator manual for more information.	
RCU_Squelch	Type	Default Value
	uint8	255
	Description: Set the live radio squelch level. Values in the range 0 - 255 are valid, with 255 being the highest level.	

Control Inputs		
RCU_Power-Level	Type	Default Value
	uint8	0
	Description: Set the live radio power level. There are three possible settings: 0 – LO 1 – MED 2 – HI Some settings are not possible in all cases; refer to the radio's operator manual for information on setting the power level.	
RCU_TextMode	Type	Default Value
	uint8	0
	Description: Set the live radio text mode. There are two possible settings: 0 – PT (plain text) 1 – CT (cipher text)	
RCU_ModMode	Type	Default Value
	uint8	0
	Description: Set the live radio receive modulation mode. There are two possible settings: 0 – AM 1 – FM Some settings are not possible in all cases; refer to the radio's operator manual for information on setting the modulation mode.	

Control Inputs		
RCU_ TXMod- Mode	Type	Default Value
	uint8	0
	Description: Set the live radio transmit modulation mode. There are two possible settings: 0 – AM 1 – FM Some settings are not possible in all cases; refer to the radio's operator manual for information on setting the modulation mode.	
RCU_ Operat- ing Mode	Type	Default Value
	uint8	0
	Description: Set the live radio's operating mode. There are four possible settings: 0 – Bypass 1 – Receive 2 – Transmit 3 – Beacon In <i>Bypass</i> mode, the component does not attempt to control the operating mode. This is useful if there is another entity driving the live radio's operating mode; for example, if an operator were to use a handset with the radio – while the remote control system is connected – one would prefer to set <i>Bypass</i> so that the Refer to the radio's operator manual for information on the other three modes. Note: <i>Transmit</i> and <i>Beacon</i> mode implies that the transceiver will be actively keyed and transmitting over the air. Take caution when using these; the radio should have an antennae or equivalent load available.	

Control Outputs		
Interface	Type	Default Value
	string	none
	Description: A helpful status message about the state of the live radio remote control interface. Some of the messages one might see are: Remote Control Off: Signifies that the serial messaging is disabled for the interface (most likely because ControlEnable is False) Err: Bad device name / chan: An invalid ACE-RIU device or channel has been specified. Err: check mask / log: A control query might be failing consistently, check component log (see the radio.log_level control) for more information. Okay: Normal operation	
Success- Mask	Type	Default Value
	uint64	0
	Description: A 64-bit mask for tracking the outcome of commands to the live radio. Refer to the <i>Command List</i> to locate specific commands in the mask. When a command succeeds (as determined by receiving an ACK from the live radio and or receiving a valid response) the bit for the command is set to 1. The bit is set to 0 when the command is sent again.	

Control Outputs		
Error-Mask	Type	Default Value
	uint64	0
	Description: A 64-bit mask for tracking the outcome of commands to the live radio. Refer to the <i>Command List</i> to locate specific commands in the mask. When a command fails three times (as determined by receiving a NAK from the live radio, or by receiving a response that the component does not understand – which as of this writing means anything not in the protocol defined by in the General Dynamics URC-200 V2 Manual, Document No. 99-P42304K) the corresponding bit in the flag is set to 1. If the same command succeeds just once afterwards that bit is set back to 0.	
Option	Type	Default Value
	string	0
	Description: A string detailing the options installed on the URC-200 transceiver. Valid strings include: None: No options installed. 30_90: The EBN-30 option is installed. 420: The EBN-400 option is installed. 30_90 && 420: Both EBN-30 and EBN-400 options are installed. Unavailable: Displayed on error.	
Squelch Status	Type	Default Value
	int8	0
	Description: The live radio's squelch status, two values are valid: 0 – The transceiver is squelched 1 – The transceiver's squelch has broken -1 is shown on error	

Control Outputs		
Overtemp	Type	Default Value
	int8	0
	Description: The live radio's overtemp status, two values are valid: 0 – Temperature is okay 1 – Overtemp Condition -1 is shown on error	
Synth-Lock	Type	Default Value
	int8	0
	Description: The live radio's synthesizer lock status, two values are valid: 0 – Synthesizer is unlocked 1 – Synthesizer is locked -1 is shown on error	
SWVersion	Type	Default Value
	string	None
	Description: The live radio's software version; 'Unavailable' is shown on error.	
Preset	Type	Default Value
	int8	0
	Description: The live radio's preset number, valid values are in the range 0-9. -1 is shown on error.	
Freq	Type	Default Value
	int64	0
	Description: The live radio's receive frequency. -1 is shown on error.	

Control Outputs		
TxFreq	Type	Default Value
	int64	0
	Description: The live radio's transmit frequency. -1 is shown on error.	
Squelch	Type	Default Value
	int16	0
	Description: The live radio's squelch level, valid values are in the range 0-255. -1 is shown on error.	
Power-Level	Type	Default Value
	int8	0
	Description: The live radio's power level. Valid values are: 0 – LO 1 – MED 2 – HI -1 is shown on error	
TextMode	Type	Default Value
	int8	0
	Description: The live radio's text mode. Valid values are: 0 – PT (plaint text) 1 – CT (cipher text) -1 is shown on error	

Control Outputs		
ModMode	Type	Default Value
	int8	0
	Description: The live radio's receive modulation mode. Valid values are: 0 – AM 1 – FM -1 is shown on error	
TxMod-Mode	Type	Default Value
	int8	0
	Description: The live radio's transmit modulation mode. Valid values are: 0 – AM 1 – FM -1 is shown on error	
OperatingMode	Type	Default Value
	int8	0
	Description: The live radio's operating mode. Valid values are: 1 – Receive 2 – Transmit 3 – Beacon -1 is shown on error	
RxSignal-Strength	Type	Default Value
	int16	0
	Description: The live radio's receive signal strength, valid values are in the range 0-255. -1 is shown on error.	

Internal Parameters		
Device-Name	Type	Default Value
	device_id	n/a
	Description: Select the name of the ACE-RIU device.	
Device Channel	Type	Default Value
	riu_channel	n/a
	Description: Select the ACE-RIU serial channel; serial channels can be either A or C.	
Radio Handle	Type	Default Value
	string	n/a
	Description: Enter a useful handle to help identify this radio object. The handle is useful when using the RMS Remote Control interface.	
radio.log_level	Type	Default Value
	uint32	0
	Description: Set this parameter to affect how much logging the component outputs. As of this writing, only a level of 1 is supported. It is recommended that this parameter be set to 0 if one is not actively debugging the component or the live radio interface as spurious log output will be produced. Looking at the log is useful for identifying specific commands that are unresponsive. Note: This parameter is only accessible through the full view.	

Appendix A: Component Revision Notes

This appendix details when new components are introduced, obsoleted or significantly changed.

Updated components include (12/11):

- Audio/Playsound
- Control/Counter
- Radio/RCUbasic
- Radio/Transceiver

Software version **4.33** (09/11) and **4.34** (11/11)

Updated components include:

- Audio/Demux

New components include:

- Audio/Pulse Sequence

Software version **4.32** (07/11)

Updated components include:

- Audio/Playsound

Software version **4.31** (04/29/11)

New components include:

- CommPanel/CommPanel8HRTF4
- Audio/HRTFOut4
- Remote Control/URC-200

Software version **4.30** (02/23/11)

New components include:

- Audio/ByteSplitter
- Control/PulseStep

Software version **4.29** (11/22/10)

Updated components include:

- EnvCue/5BandFilter

Software version **4.28** (09/24/10)

Updated components include:

- Radio/Relay

Software version **4.26** (05/24/10)

New components include:

- Audio/PulseStream
- Control/PassThrough

Updated components include:

- IOInterfaces/AmpOut
- Control/Incrementer

Software version **4.24** (03/22/10)

Updated/Fixed components include:

- Transceiver
- VORTAC Controller
- RCUBasic

Software version **4.23** (01/27/10)

New components include:

- Audio/SimpleMixer
- CommPanel/CommPanel8Stereo
- Dynamics/AGC, CompressorLimiter, Expander, and Gate
- Radio/ColocatedBeacon

Updated/Fixed components include:

- Audio/Playsound
- Radio/MorseKeyer
- Radio/MarkerTone

Software version **4.22** (11/24/09)

Updated/Fixed components include:

- Audio/Wave
- Radio/MorseKeyer
- Audio/Delay
- Audio/Compressor
- Radio/Transceiver

Software version **4.21** (10/29/09)

New components include:

- Control/Delay
- Program Specific component

Updated/Fixed component include:

- Audio/Compressor
- Audio/AutoDred
- Audio/Mixer
- IOInterfaces/ACU2
- Radio/Satellite

Software version **4.20** (09/18/09)

New components include:

- EnvCue/5BandFilter
- EnvCue/MultiFilter
- IOInterfaces/ACE_RIU_SerialByteOut
- IOInterfaces/ACU2_SerialByteOut
- IOInterfaces/VoisusChannel
- Radio/HFServer

Updated/Fixed components include:

- Audio/Compressor

Software version **4.19** (08/04/09)

New components include:

- Audio/ NoiseSource
- Speech/ TTS
- Program Specific Components

Updated/Fixed components include:

- Audio/ AutoDRED
- Audio/ MessageList
- Record/ Replay
- Radio/ Satellite
- Radio/ Transceiver
- Radio/ Relay

Software version **4.18** (06/30/09)

New components include:

- Audio/ Delay
- Control/ MessageList updated and moved to Audio group
- IOInterfaces/ ACU2channel
- IOInterfaces/ RadiusChannel
- Radio/ Satellite
- Program Specific Components

Software version **4.16** (04/27/09)

New components include:

- Audio/ Compressor
- Internal Testing Components
- Program Specific Components

Software version **4.15** (03/13/09)

New components include:

- Audio/ Demux
- Audio/ Envelope
- Audio/ PEnvelope
- Control/ Incrementer
- FDACP (for a specific program)

Software version **4.14** (02/26/09)

New components include:

- Control/ ByteToBit
- Control/ ByteMerger
- HostControl/ CellIn
- HostControl/ CellOut
- Environmental Cue/ VibrationCapture
- Radio/ Intercom Transceiver

Updated components include:

- Comm Panel Components for optimization

Fixed components include:

- Audio/ Pfilter
- Audio/ Auto Dred for gain defaults

Software version **4.13** (01/26/09)

Fixed components include:

- Fixed audio pops on Transceiver receive side.

Software version **4.12-4** (11/26/08)

Updated components include:

- Control/ MessageList to support index repeat.

Software version **4.12-1** (10/31/08)

Updated components include:

- Comm Panel component for signal gain effects on sidetone.
- The Audio/ Record Replay component for timing.
- Improved the Intercom/ IcomTx component.

Software version **4.11-5** (10/14/08)

New components include:

- Platform/ RelativePosition

Updated components include:

- Comm Panel component

Software version **4.11-2** (09/16/08)

Updated components include:

- Environmental Cue/ PropRotor component

Fixed components include:

- Platform/ RelativePosition

Software version **4.10-4** (09/04/08)

Fixed components include:

- Audio/ Filter

Software version **4.10-3** (08/22/08)

New components include:

- Environmental Cue/ PropLevelD component which was renamed to Prop Rotor in a later version

Software version **4.8-1** (06/04/08)

New components include:

- Radio/ ICU
- Radio/ Transmitter
- Radio/ Receiver
- AudioIO/Headphone3DOut

Software version **4.7-1** (05/19/08)

New components include:

- Radio/ TransmitterRCU
- Network Intercom RCU components

Software version **4.6-1** (04/16/08)

New components include:

- Audio/ Pfilter

Fixed components include:

- Host Control/ HostOut component for proper bit handling
- IOInterfaces/ RIUchannel component for displaying the correct number of channels

Software version **4.4-4** (04/02/08)

Fixed components include:

- Fixed the Comm Panel/ Comm Panel 16 component

Software version **4.4-3** (03/19/08)

Updated components include:

- Improved various HLA components
- DACS component aliases

Fixed components include:

- Host Control/ HostIn component to pass message variables to a link

Software version **4.2-2** (09/18/07)

Updated components include:

- Comm panel components for default operation with the PTT

Fixed components include:

- Audio/ Playsound
- Audio/ Sequencer

Appendix B: Component Credit Allocation

See Application Note 85: Telestra 4 ACE Studio Modeling Credits System on the ASTi web site at www.asti-usa.com.